(2)

ADST/WDL/TR-92-003010

Strawman Distributed Interactive Simulation Architecture Description Document Volume II: Supporting Rationale Book II: DIS Architecture Issues

Loral Systems Company
ADST Program Office
Orlando, Florida

31 March 1992

Prepared for

PMTRADE
Program Manager - Training Devices
Orlando, Florida

ADST/WDL/TR--92-003010

Strawman Distributed Interactive Simulation Architecture Description Document Volume II: Supporting Rationale Book II: DIS Architecture Issues

Loral Systems Company
ADST Program Office
12433 Research Parkway
Orlando, Florida 32826

31 March 1992

Contract No. N61339-91-D-0001
CDRL A002

Prepared for

PMTRADE
Program Manager - Training Devices
Naval Training Systems Center
12350 Research Parkway
Orlando, Florida 32826-3275

LORAL
Systems Company

REPORT DOCUMENTATION PAGE

Form approved OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	03/31/92	Version 0 03/91 -03/92

4. TITLE AND SUBTITLE
Strawman Distributed Interactive Simulation Architecture Description Document Volume II: Supporting Rationale, Book II: DIS Architecture Issues

5. FUNDING NUMBERS
Contract No. N61339-91-D-0001
CDRL A002

6. AUTHOR(S)
Beaver, Ray; Brasch, Randy; Burdick, Chuck; Butler, Brett; Downes-Martin, Stephen; Eller, Tim; Ferguson, Bob; Gotuby, Dave; Huber, Joe; Katz, Warren; Miller, Bill; Morrison, John; Sherman, Richard; Stanley, Walt; Yelowitz, K.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Loral Systems Company
ADST Program Office
12443 Research Parkway, Suite 303
Orlando, FL 32826

8. PERFORMING ORGANIZATION REPORT NUMBER
ADST/WDL/TR-92-003010

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Project Manager for Training Devices PM TRADE
Naval Training Systems Center
12350 Research Parkway
Orlando, FL 32826-3275

10. SPONSORING ORGANIZATION REPORT

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The DIS architecture defines a time and space coherent representation of a virtual battlefield environment, measured in terms of the human perception and behaviors of warfighters interacting in free play. It provides a structure by which independently developed systems may interact with each other in a well managed and validated combat simulation environment during all phases of the development process. A fundamental objective of the DIS architecture is to provide a blueprint to guide development of a general purpose system meeting the needs of a wide range of users. The architecture must therefore support the broadest possible range of user needs. At the same time, the architecture and the associated standards must provide sufficient design definition to achieve the goal of transparent interoperability of a very wide range of simulators, simulations, and actual equipment operating on instrumented ranges.

14. SUBJECT TERMS

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	17. SECURITY CLASSIFICATION OF THIS PAGE	17. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

1. Network Topology

BDS-D's goal is to provide space-time coherent battlefields to its users. BDS-D proposes a two-tier architecture to meet this goal. Section 3, Volume I of this document presents the architecture, and Section 5 of Volume I discusses related network issues.

This section presents the rationale behind this two-tier architecture. Section 1.1 presents the architecture. Section 1.2 lists the requirements the BDS-D hardware and software components must meet in order to successfully realize the architecture. Section 1.3 lists the evaluation criteria for each potential hardware and software component.

1.1 DIS Architecture

There are two levels of abstraction that help to illustrate the architecture. This section presents each, and explains how each satisfies or highlights architecture requirements. These levels are:

- The DIS Exercise Level, which highlights functionality and performance requirements for the underlying virtual networks.
- The Virtual Network Level, which presents the two-tier architecture, and explains how it can help achieve the required levels of functionality, performance, interoperability, and security.

1.1.1 DIS Exercise Level

1.1.1.1 Multiple Simultaneous Exercises

DIS must support multiple simultaneous exercises, shown in Figure 1-1. Each DIS exercise (e.g., the top layer in the figure) is a space-time coherent representation of the virtual battlefield. Each DIS exercise is made up of DIS entities and Simulation Support entities. DIS entities are such things as tanks and helicopters, and it is a stated goal of DIS to support upwards of 10,000 in any one exercise. Simulation Support entities are such things as Data Loggers and After-Action Review facilities. These entities interact via the "DIS PDU Standard," defined in draft form in "Protocol Data Units for Entity Information and Entity Interaction in a Distributed Interactive Simulation."

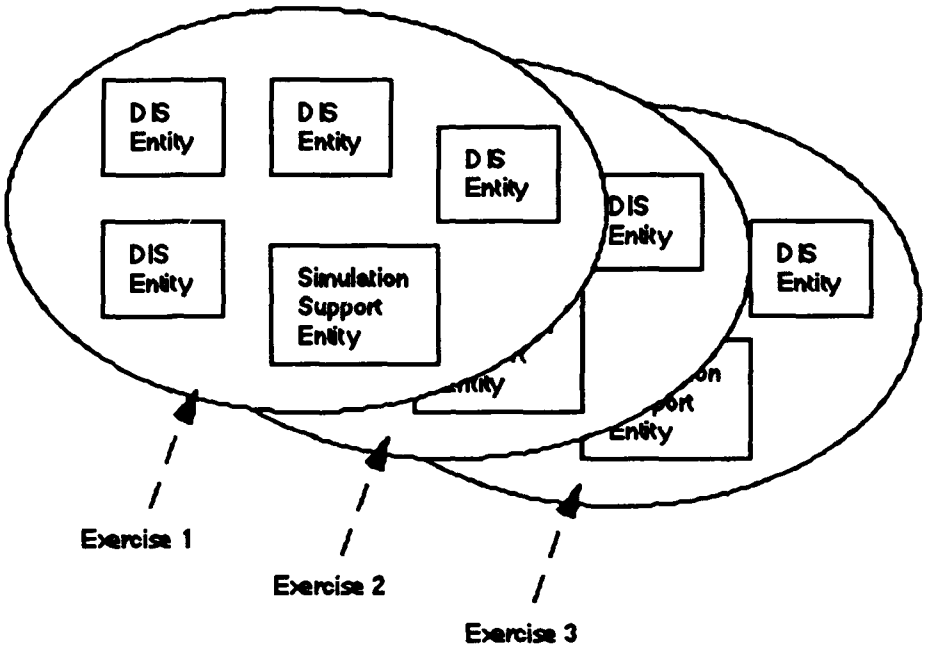


Figure 1-1. DIS virtual networks support multiple exercises.

1.1.1.2 Per-Exercise Fidelity

BDS-D provides flexibility on a per-exercise basis by defining an exercise in terms of several databases (Volume I, Sections 3 and 4):

- The SIMWORLD Database
- The BATTLEFIELD Database
- The SESSION Database

Together, these databases specify the exercise's desired fidelity. Different levels of fidelity may require different levels of performance from the underlying system. Volume II, Book I entitled "Time/Space Coherence and Interoperability" explains how fidelity is quantitatively related to system performance. Thus, on a per-exercise basis, BDS-D is required to:

- Deliver DIS PDUs within a given maximum latency, and a given maximum latency variance.
- Deliver a certain number of DIS PDUs per second, which is a function of the number of DIS entities in the given exercise.
- Deliver DIS PDUs with a drop rate below a given maximum, and with a bit error rate below a given maximum.
- Ensure secure operation at a given security classification level.

1.1.2 Virtual Network Level

This section will present the two-tier architecture for DIS exercises, and will show how the architecture fulfills the requirements listed above.

1.1.2.1 DIS Exercise as Virtual Broadcast Network

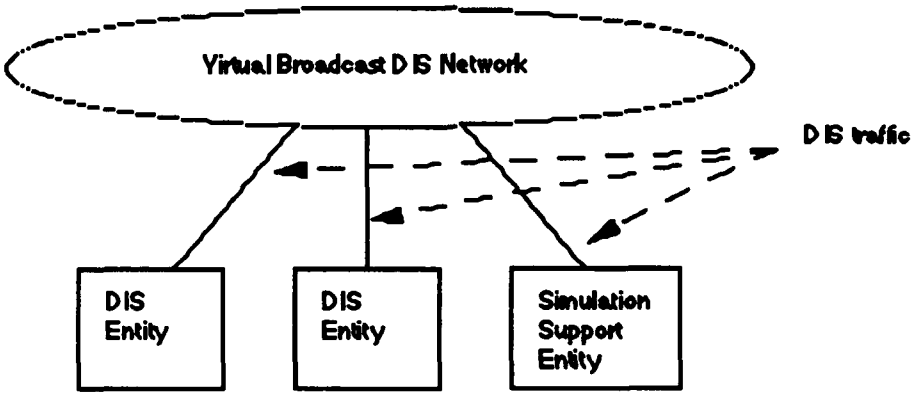


Figure 1-2. DIS exercise as a virtual broadcast network.

Figure 1-2 shows how a given exercise can be thought of as a virtual broadcast network, whose nodes are DIS entities and Simulation Support Entities. A virtual broadcast network can be thought of as a "community of interest," in which packets sourced by any member of the community are routed to all members of the community. I.e., DIS PDUs are broadcast to all the DIS entities.

In order to understand the following discussions, it is important to understand that *DIS entities have "identifiers," not "addresses."* For example, a Detonation PDU sent "from" entity1 "to" entity 2 is in fact *routed to all the entities in the exercise*, so that they may display the impact. Thus, entity identifiers, even though they contain source-host pairs, are not addresses which are used for routing.

1.1.2.2 Two-Tier Architecture: Cells as Virtual Broadcast Networks

Although the above single-tier virtual broadcast network is appealing in its simplicity, several BDS-D goals argue for grouping DIS entities into "cells," each of which is a virtual broadcast network as shown in Figure 1-3 below:

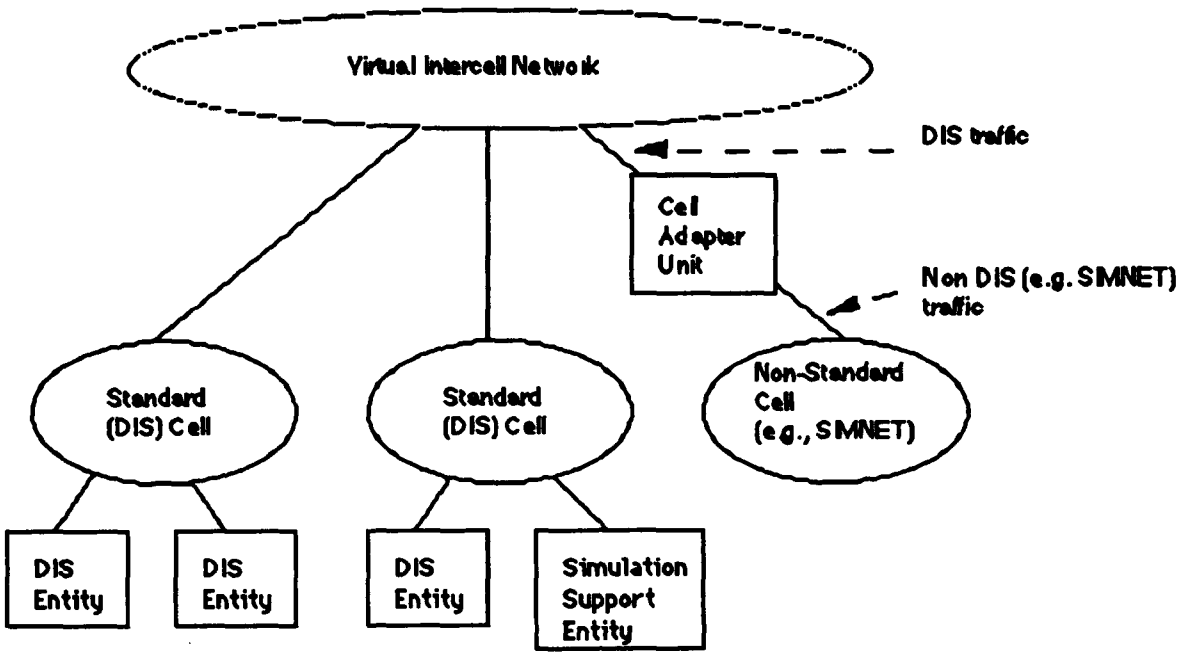


Figure 1-3. Upper tier virtual network connects multiple cells in a single exercise.

1. Interoperability between DIS and non-DIS (e.g., SIMNET) simulators. If the simulators do not share a common protocol, a translator/gateway becomes necessary. This results in a partition into two networks, with a "Cell Adapter Unit" (CAU) acting as a protocol translator.[1]

2. World-wide interconnection of simulators. The two-tier architecture is intended to be analogous to a WAN/LAN architecture, with the Virtual Intercell Network as a WAN, and the cells as LANs. There is no reason for the "virtual broadcast" mechanisms in each tier to be common, shared, or even the same. In fact, the intercell network may elect to use virtual circuit techniques, and the cell networks may elect to use broadcast-based techniques.

1.1.2.3 Two-Tier Architecture Supports Bandwidth Conservation, Security

The desire for security in inter-entity communications, and the desire to conserve intercell and cell network bandwidths can also be further addressed by the two-tier architecture.

---

[1] Note that this is essentially an $O(n)$ problem, where n is the number of entities. (An "order n" problem grows linearly with increasing n.)

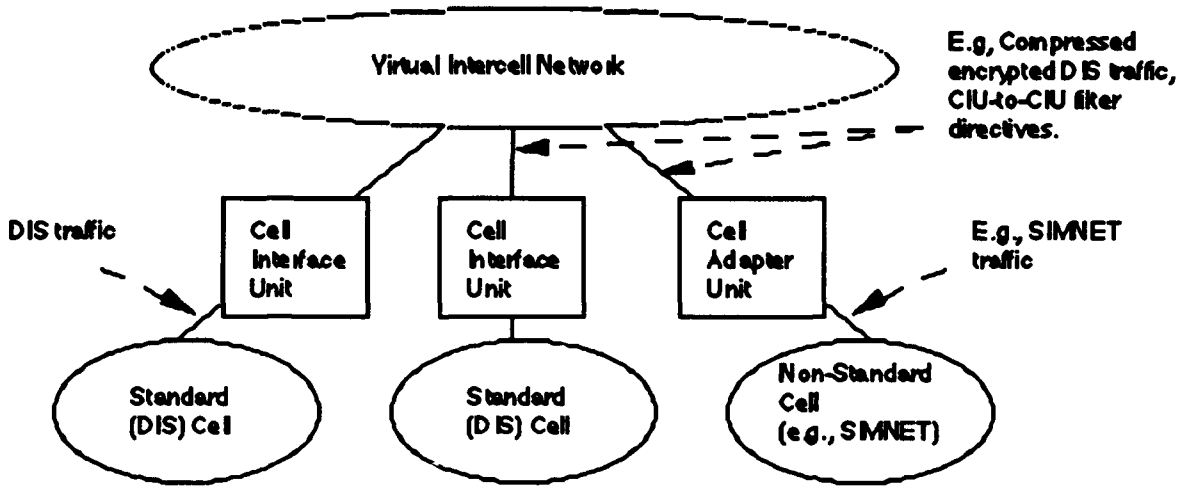


Figure 1-4. The two-tiered architecture supports bandwidth conservation and security.

Figure 1-4 shows an exercise's virtual network divided into cell virtual broadcast networks. The partitions are bridged by Cell Interface Units (CIUs). CIUs are logical DIS-level gateways.[2] (Note that CAUs are CIUs that also perform protocol translation, and that in the ensuing discussions, CIUs include CAUs unless otherwise noted.) CIUs can:

- Provide a ring of security around the untrusted virtual intercell network. The CIUs can make content-based forward/no-forward decisions, and/or they can perform encryption/decryption.

- Manage traffic flows across the cell and intercell networks. Following the SIMNET long-haul gateway example, CIUs can compress DIS packets based on content, in order to conserve intercell bandwidth.[3] They can perform content-based routing decisions, capitalizing on dynamic multicast capabilities of the intercell network to preserve both cell and intercell network bandwidths. (Note that these schemes may require as yet unstandardized, non-DIS, CIU-to-CIU communication.)

---

[2] Another *O(n)* problem.

[3] Content-based compression can arguably achieve a higher compression than simple compression schemes. For instance, the Entity State PDU constitutes the bulk of the traffic. One could create a highly-compressed Entity State PDU by mapping the 48-bit Entity IDs to 16-bit integers, and by eliminating the unchanging information (protocol version, exercise id, three padding fields, force ID, both entity types, entity appearance, entity marking, capabilities, and number of articulation parameters), saving 49 bytes out of a minimum 144 bytes on each Entity State transmission. This would result in approximately a 34% bandwidth savings. Further, any compression performed by the WAN is transparent to the CIUs.

- Promote interoperability by arbitrating difference in databases between cells of different fidelity.[4]

1.2 DIS System Requirements

This section lists the requirements for the BDS-D hardware/software suite for network components as shown in Figure 1-5.. The requirements are drawn from the above analysis, and from other indicated sections of this document. They are listed in the following subsections, in approximately decreasing order of importance.

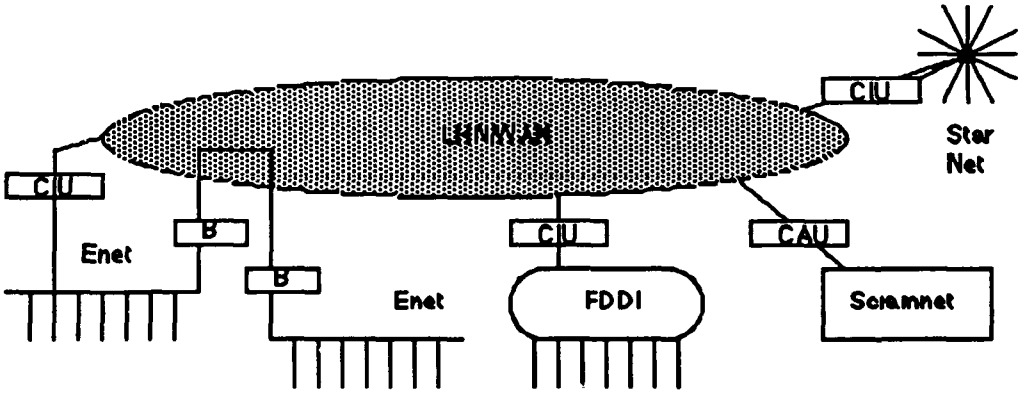


Figure 1-5. The virtual network supports standard LANs and WANs.

1.2.1 Performance

Performance is more important than the sum of all other requirements. An interoperable, secure, reconfigurable, cheap solution that fails to meet the performance goals is worse than useless because it won't work, but it still costs money.

Specific performance requirements, in approximately decreasing order of importance:

- Ability to support multiple, nested virtual broadcast networks
- High Throughput, both absolute bandwidth and number of packets per second
- Low Latency Variance
- World-wide Interconnection
- Low Latency

---

[4] Yet another *O(n)* problem.

- Low drop rate
- Low bit error Rate
- High Reliability and System Availability.

Virtual broadcast is also called an *n-way* multicast. For small *n,* which is the number of nodes on the virtual broadcast net, *n-way* multicast can be built out of *n 1-way* multicasts, in which packets from one specific member (hence *1-way)* are routed to all the others.

Multicasting does not require a connectionless rather than a connection-oriented hardware/software approach. Multicast support is not limited to a specific (OSI) level in the protocol stack. It does not require a specific protocol stack. Furthermore, the WAN and LAN multicast techniques are not required to be, nor will they likely be, the same. For example, a connection-oriented multicast can be performed between geographically distributed sites (based on exercise ID), yet the individual simulation entities on a LAN at a site communicate PDUs in a connectionless multicast manner (for different exercises).

The average rate of DIS PDUs and throughput as well as the short term peak PDU rates and throughput sourced as a function of the number of simulation entities are extrapolated from SIMNET experience and DIS packet sizes.

Table 1-1. Average long term and peak short term sourced DIS PDU rates and throughputs.

Number of Simulation Entities	Average PDU Rate (PDUs/sec)	Average Throughput (Mbps)	Peak PDU Rate (PDUs/sec)	Peak Throughput (Mbps)
1,000	1,000	1.5	2,500	3.75
10,000	10,000	15	25,000	37.5
100,000	100,000	150	250,000	375

This is especially important for the intercell network, which will have to route the sum of the traffic from several simultaneous exercises. The extremely large number of packets to be routed at each cell interface means that the per-packet routing service time must be vanishingly small.

Virtual network latency or delay is shown for a virtual network connection for two simulators, two LANs, and a commercial WAN. Routers on each LAN interface to the common WAN. Table 1-2 shows average, one standard deviation, and 95th percentile virtual network latencies for OSI protocol stacks (TP4, CLNP, LLC) for PDUs averaging 1500 bits, and 80% occupancies of communications channels for different protocol processing CPU MIPS at each stage, i.e., simulator, router, WAN. The latency models are extrapolated from data provided

by the National Institute of Standards and Technology (NIST) for OSI protocol stacks using 0.8 MIPS processors. The queuing delays assume a M/M/1 model.

Table 1-2. Virtual network latencies for OSI protocol stacks versus CPU MIPS.

CPU MIPS	Average Latency (msec)	One Standard Deviation Latency (msec)	95th Percentile Latency (msec)
1	500	500	1500
10	50	50	150
25	20	20	60
50	10	10	30

The number of sites supporting a 1,000 simulation entity exercise in BDS-D Phase 1 is expected to be two to three. A 10,000 entity exercise in BDS-D Phase 2 would be distributed over 20 to 50 sites. A 100,000 entity exercise in BDS-D Phase 3 is anticipated to be distributed over 50 to 200 sites. World-wide connectivity over international commercial WANs is achieved via the interoperability of the OSI protocols based on international agreements through the Consultative Committee on International Telecommunications (CCITT), a body sponsored by the United Nations (UN).

PDU network (LANs and WANs) latency or delay is anticipated to have a mean of 50 msec with 95% of the PDUs serviced in less than 150 msec. This model includes queueing delays for network and processing occupancies of 80%. The system drop rate (excluding LAN contention) due to switch buffer overflow is anticipated to be less than one in 10^7 PDUs. The bit error rate is anticipated to be less than one in 10^7 bits. Commercial fiber-optic WANs today experience error rates of less than one in 10^{11} bits. Network system (LAN/WAN hardware and software) availability is anticipated to be 99.9% based on objectives set by LAN vendors and WAN Interexchange Carriers (IXCs).

1.2.2 Interoperability

Interoperability is a prime goal of DIS, because it benefits the customer by increasing capability and reducing cost.

Interoperability exists at more than one level in the two-tier architecture. The simulator-to-simulator interface has already been addressed by the DIS Draft Standard. Interfaces that require further study include the CIU-to-CIU interface,

which includes any bandwidth-reduction protocols, and the CIU-to-Intercell Network interface.

Interoperability can be achieved through the use of Open, multi-vendor, and COTS standards based on OSI protocols from layers 1 through 3, subject to performance requirements for throughput and latency.

1.2.3 Security

BDS-D is required to conduct secure exercises at the level of DoD Secret and Special Access Programs (SAP). See Section 7 of Volume I.

1.2.4 Reconfigurability of N-way or 1-way Multicast Groups

Although dynamic *n-way* multicast is desirable, a per-exercise configuration of cells specified in the Session Database is sufficient . Minimum advance notice for network reconfiguration is (TBD).

Guidance from the Defense Modeling and Simulation Office (DMSO) suggests that virtual network configuration should be changeable on the order of one hour warning notice before an exercise.

1.2.5 Cost

Annual recurring costs of the DIS WAN are estimated using three different methods offered by commercial IXCs. The first and least expensive method is a private network of leased dedicated lines between DIS locations. A two-tiered tandem arrangement is assumed to lower the cost estimate compared to a fully interconnected network of (n)(n-1)/2 links. Costs are driven by the average link distances in airline miles between sites. Even though this method is theoretically the least expensive, it assumes that the connectivity matrix between DIS sites is static. In reality, this assumption is not true and there is often a high churn rate (50% annually in commercial private line networks) with resultant installation costs and loss of long term discount options.

The second method is the flat rate basis of commercial frame relay services that connect routers on LANs to the local IXC Point of Presence (POP). The PDUs are then transported over a shared backbone network with minimum bandwidth-delay guarantees and the capability to burst above the reserved bandwidth to the remaining bandwidth of the facilities at no extra cost.

This method is slightly more costly than a static private line network yet more robust to the dynamic connectivity matrix between sites and surges in PDU rates during exercises.

The third and most expensive method is based on the IXCs charging by the packet. In this estimate, the IXCs charge not on the basis of sourced PDUs, but

rather delivered (multicast) packets and this explains the high cost relative to private networks and frame relay.

Figure 1-6 shows the cost estimates for the three types of communications for 1,000 simulation entities used 40 hours per month. The entities are geographically distributed over 5 to 50 sites.

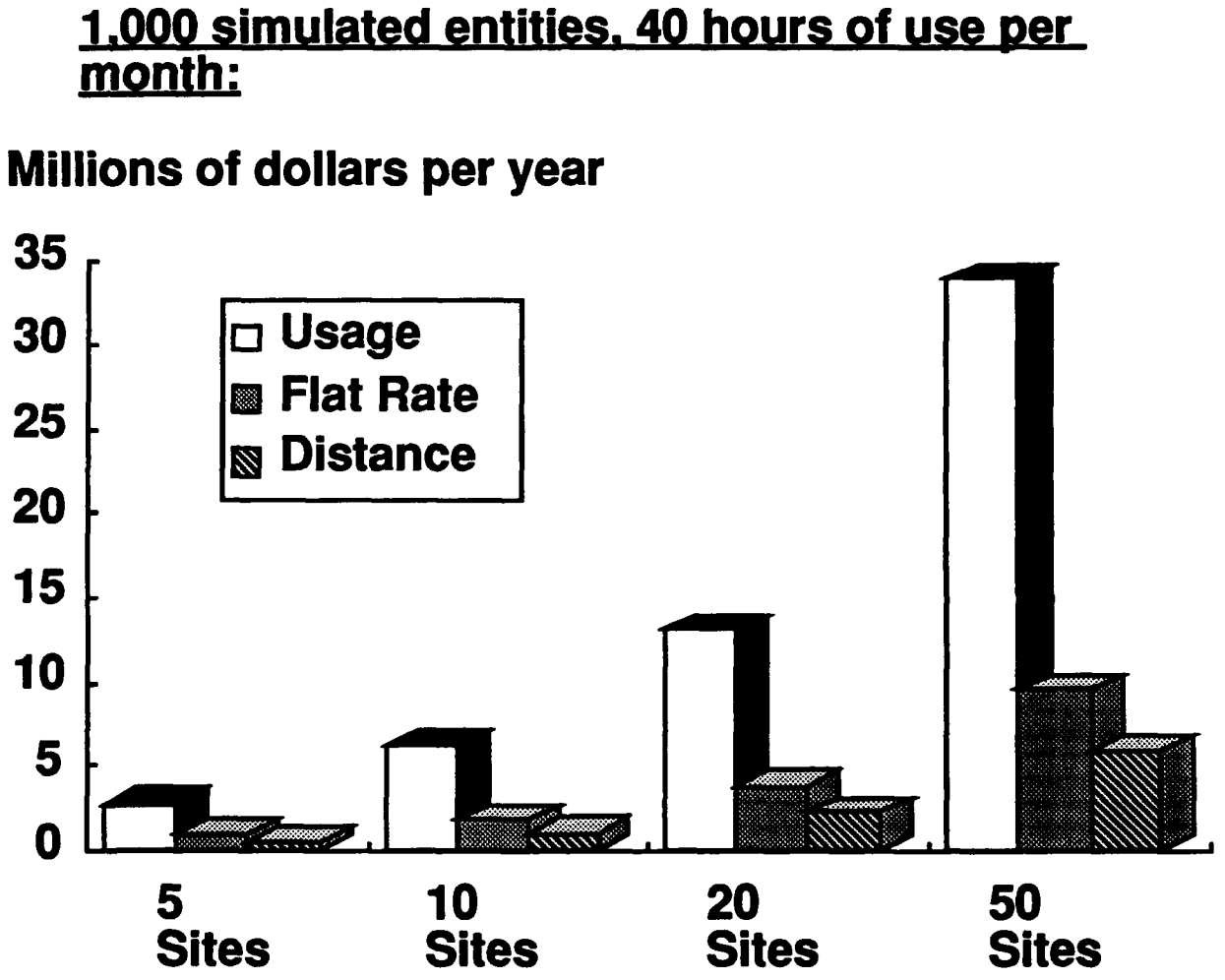


Figure 1-6. Estimated communications costs for 1,000 entities and 40 hours use per month.

Figure 1-7 shows the cost estimates for the three types of communications for 10,000 simulation entities used 40 hours per month. The entities are also geographically distributed over 5 to 50 sites.

10,000 simulated entities, 40 hours of use per month:

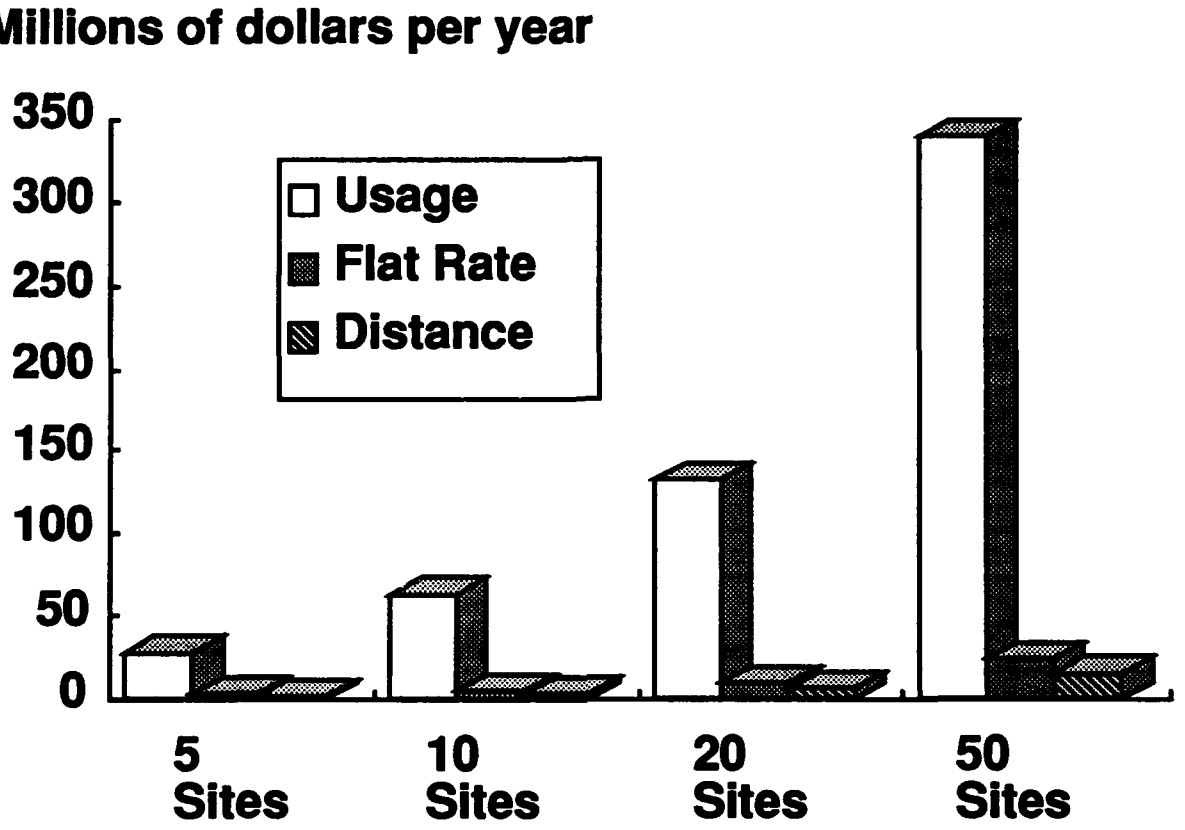


Figure 1-7. Estimated communications costs for 10,000 entities and 40 hours use per month.

1.3 Architecture Component Evaluation

1.3.1 Components to be Evaluated

Figure (above, physical) shows the BDS-D components to be evaluated:

- WAN Hardware, and Software
- LAN Hardware, and Software
- Internetwork Protocol Stacks
- CIUs
- CAUs

1.3.2 Network Management

This section has two goals. The first goal is to explain what has been done by committees within the International Standards Organization (ISO) to define and organize the services required to perform communications resource management. Concepts and terminology relative to the management of communications resources are introduced. These resources are assumed to be organized within the framework of the Open Systems Interconnection (OSI) Reference Model.

Section 1.3.2.1 contains an introduction to OSI network management with an in-depth description of the OSI Systems Management Model.

Section 1.3.2.2 is a presentation of the current state of network management standards in 1991.

1.3.2.1 OSI Management

ISO standards committees employ an abstract model, the System Management Model, to organize the services offered by an OSI compliant network management system. Specific services and protocols are defined in related protocol specification standards. This section introduces the concept of OSI management and describes the Systems Management Model.

The first draft of the OSI Reference Model was introduced in 1978. Since then several extensions have been incorporated into the basic model. These extensions provide additional functions used to facilitate information transfer within a large network. One of these is the OSI Network Management extension. The OSI Management Environment is defined as "that subset of the total OSI Environment which is concerned with the tools and services needed to control and supervise interconnection activities and managed objects". A managed object could be a piece of hardware, a software component or a collection of information such as a database. The object does not need to be an OSI resource, as it can fall outside of the framework established by the Reference Model. For example, a Protocol Data Unit (PDU) Manager for DIS can be defined as a managed object outside the framework of the OSI model.

ISO management standards address both the syntax and semantics of the information required to accomplish the resource management. They also specify the communications services required to transport this information within an OSI environment. The standards do not specify how specific management functions are accomplished. That definition falls under the domain of the user application programs.

The OSI/NM Forum is a group of 60 vendors and service providers working to accelerate OSI network management interoperability. The Forum promotes the use of existing and draft standards as the basis for interoperability specifications.

Documents produced by the Forum include a forum architecture, protocol specification, object specification framework, and application services.

Figure 1-8 depicts the organization of resource management within the OSI environment. OSI management focuses on the monitoring and control of "managed objects," where a managed object can be any resource (hardware or software).

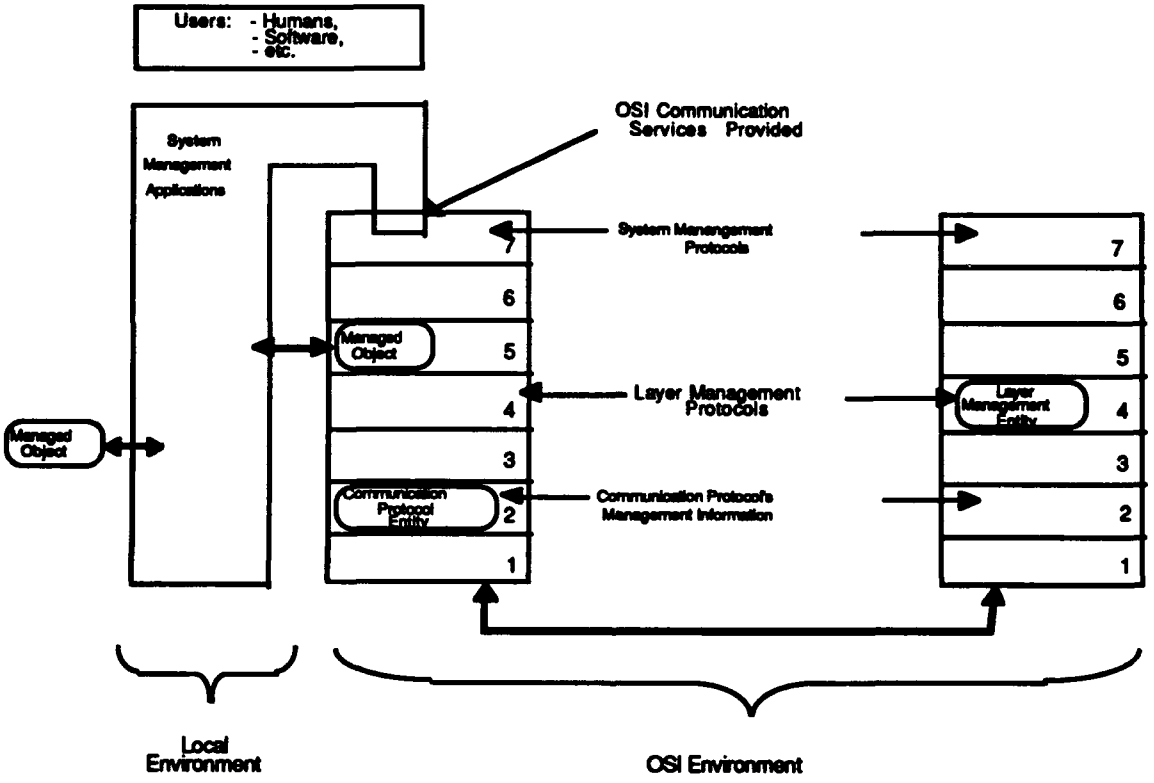


Figure 1-8. OSI management.

As shown in Figure 1-8, the systems management applications and the user's interaction with these applications fall outside of the scope of OSI management standards. Three forms of management information exchange are defined within the OSI management architecture. These are:

1. Systems Management
2. Layer Management
3. Layer Operation

Systems management is the preferred form of management information exchange. Systems management provides mechanisms for the monitoring and

control of all managed objects. Systems management is the only means by which OSI management of multiple layers is accomplished.

Layer management provides for the monitoring and control of managed objects within a given layer. Layer management protocols should only be used when either the systems management services do not support the exchange of layer management information or when the exchange is not supported by higher layer services. Layer management entities are processes, which are separate from those used to provide the communications functions. As such, layer managers can maintain logs containing parameter values related to specific communications functions such as average delays between entities. In addition layer managers can test the services provided by the layer beneath them.

Management functions within the communications protocols themselves are referred to as layer operations. They differ from layer management functions in that as soon as that instance of the protocol is not needed the layer operations no longer exist. Examples of information conveyed within the communications protocols are:

1. Error information for that particular instance of communications.
2. Parameters used to modify the protocol during that instance of communications.
3. Parameters used to control the establishment or release of a specific connection.

The things to remember are: 1) Systems management is the most general, followed by layer management and then layer operations, and 2) the preferred approach is to use systems management application processes.

The standards bodies have concentrated on the definition and standardization of the services required to support systems management processes. As is typical for OSI functions, the OSI Management Functions are defined using an abstract model in this case the Systems Management Model. In addition to the Systems Management Model some of the standards relevant to the implementation of the model have been agreed to, or suggested. You can view the model as a way to organize the various services required to support systems management while the standards define sets of service primitives used to provide the services.

Within the Systems Management Model the management of a communications environment is treated as a distributed information processing application. The interactions which take place between management processes are controlled by directives. These directives are communicated from one process to another.

Management processes are categorized as being either a managing process or an agent process. Managing processes have overall responsibility for one or more specific management activities. Agent processes do the actual object

manipulations associated with a request from a managing process. This is depicted in Figure 1-9.

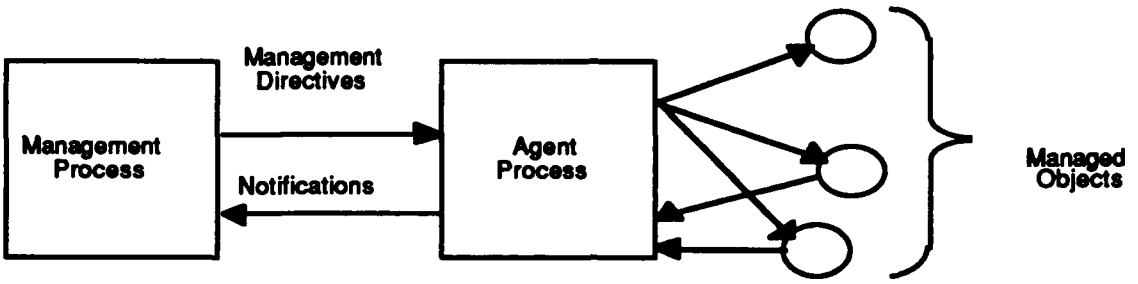


Figure 1-9. Management process interactions.

A view of the overall organization of the Systems Management Model is depicted in Figure 1-10.

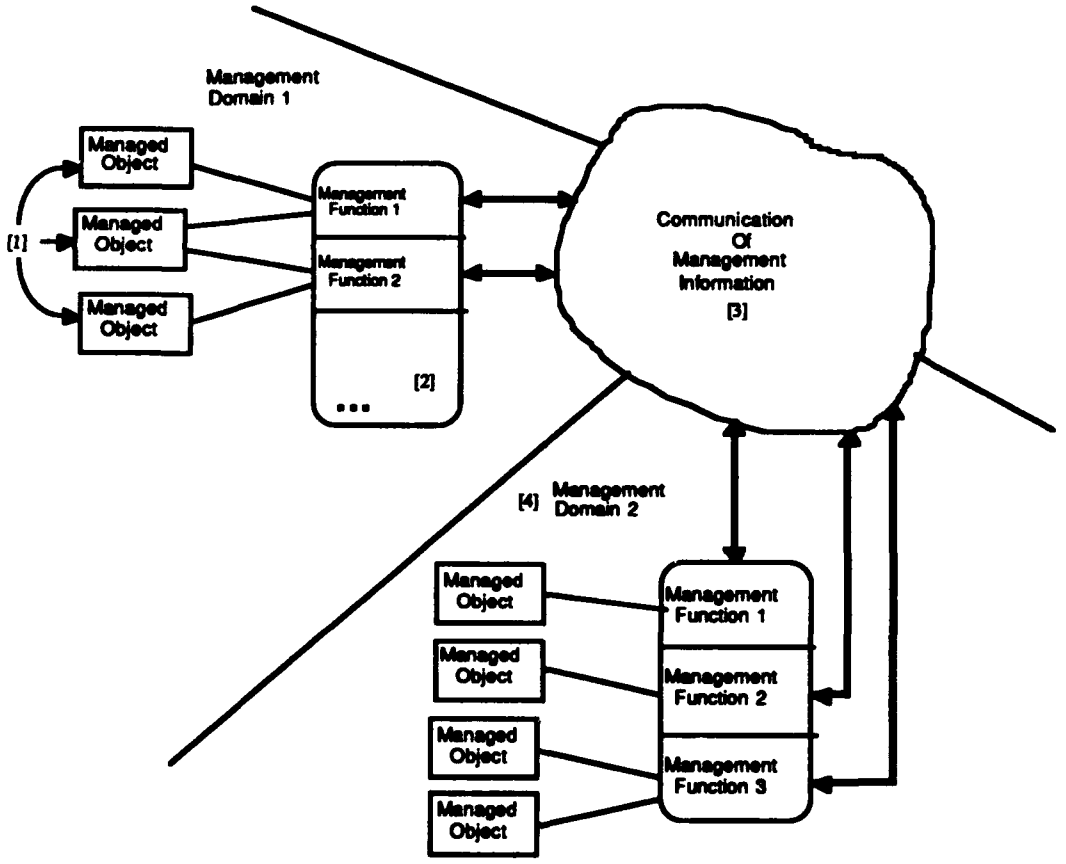


Figure 1-10. System management overview.

As shown in Figure 1-10, the model consist of four components, which describe the manipulation of management information. The first component, the

information aspect , introduces the concept of managed objects and the information used to describe these objects. Management activities are ultimately affected through the manipulation of managed objects. The second component, the *functional aspect* of the model, is the functional grouping of management activities. An example of a functional grouping is fault management. You can think of a functional grouping as a basic user control mechanism. By organizing management activities into functional groupings, the end user can exercise management control while remaining isolated from the actual details of the managed object. The third component, the *communication aspect* of the model, describes the communications services required to convey management information between management processes. Finally, the fourth component, the *organizational aspect*, deals with the organizational requirements for managing a collection of systems operating in an OSI environment. This component describes the requirements for assigning control of a management function to a chosen process and the roles of other processes (agents) in the achievement of that function.

A *managed object* is the OSI management's view of a resource that is subject to management. This abstract definition provides the framework to allow just about anything to be defined as a managed object. For example, a connection such as a SAP can be a managed object as well as a piece of software or a physical piece of communications equipment. Four features of a managed object are used to monitor and control its functions. These are:

1) The object's existence
2) The object's attributes
3) The object's states
4) The object's relationships

A managed object *exists*, if it is named in the appropriate database, and has an associated set of management information that is accessible through OSI management services. Included in the management information is the set of management operations that can be performed upon the object and the effect those operations have on the object. Characteristics of an object are specified through object *attributes*. Attributes are descriptions of selected properties of the managed object. When an object is created, a set of attributes is defined for that object. The values of these attributes may be changed but attributes cannot be added or removed. The set of managed objects (names) in a system along with their attributes constitute the Management Information Base (MIB) for that system. The model does not define how the MIB is distributed. The model dictates what information is to be included but does not restrict the way this information is distributed within the system.

The state of a managed object represents the instantaneous condition of the object's operability and availability. Managed objects may emit notifications, which are reports on their states.

Relationships define the interdependence between managed objects. Note that the model only has to say that features such as relationships exist. How the individual developer will provide the means to determine and describe these relationships will be the difficult part of the task.

When you consider the types of services that a user might require to manage a communication system, you're on the track of describing the functional areas. In ISO terminology these are referred to as the Specific Management Functional Areas (SMFAs). There are five specific functional areas defined in the model. These are:

1) Fault Management
2) Configuration Management
3) Accounting Management
4) Performance Management
5) Security Management

The goal of fault management is the detection and monitoring of abnormal network operations. Faults manifest themselves as particular events (errors) in the operation of the network. Within the OSI Systems Management Model fault management is the set of facilities used to manage error logs, accept and act on error notifications, trace faults and carry out sequences of diagnostic tests.

Configuration management provides for the identification and control of managed objects with the goal of insuring the continuous operation of communications services. Configuration management consists of the set of facilities required to initialize and close down managed objects, to collect the data necessary to determine the system's configuration state, to change the configuration of the system (switch in standby equipment) and to associate logical names with sets of managed objects.

The goal of accounting management is the determination of costs for the use of managed objects and the establishment of charges for this use. Accounting management consists of those facilities which; inform users of incurred costs, enable the fixing of account limits for the allocation of resources and provide for the combination of costs when multiple managed objects are invoked to accomplish a communications task.

Performance management is the evaluation of the long term behavior of managed objects. This differs from fault management or configuration management in that both of these tend to focus on the immediate status of a managed object such as, " is it on?", or " is a standby available? " . The information used in performance management is typically statistical data that's analyzed to determine and predict trends in the communications capabilities of the network.

The final Specific Management Function Area is that of security management. The general idea of OSI Security Management is to provide the facilities required to implement an organization's security policy, as it applies to the communications aspects of a network. Specific functions included under security management are the control and maintenance of access restrictions, the management of encryption keys and the creation and distribution of security logs, such as access audits.

A management task may require the use of services provided for under multiple specific management functions. For example, "It's broke, fix it!" would require fault determination and isolation (using fault management) and systems reconfiguration, such as changing routing tables or switching in standby equipment (using configuration management).

To reiterate, the purpose of this section is for the reader to understand how the standards organizations define OSI Network Management. The tools used are: 1) a model that organizes the required functions into conceptual groupings, and 2) systems management standards that detail how the functions are to be implemented. In describing the model we have introduced the concept of managed objects and the grouping of management functions into five general categories. A third service described by the Systems Management Model is the communication of management information.

The ability to determine the status and alter the configuration of managed objects requires the exchange of management information between cooperating systems. The communications aspect of the model defines those services which are offered by the communications layers to support the exchange of management information.

Figure 1-11 presents a global view of the communications required to support systems management functions. A user's network may include processing and communications resources that are provided by a number of vendors.

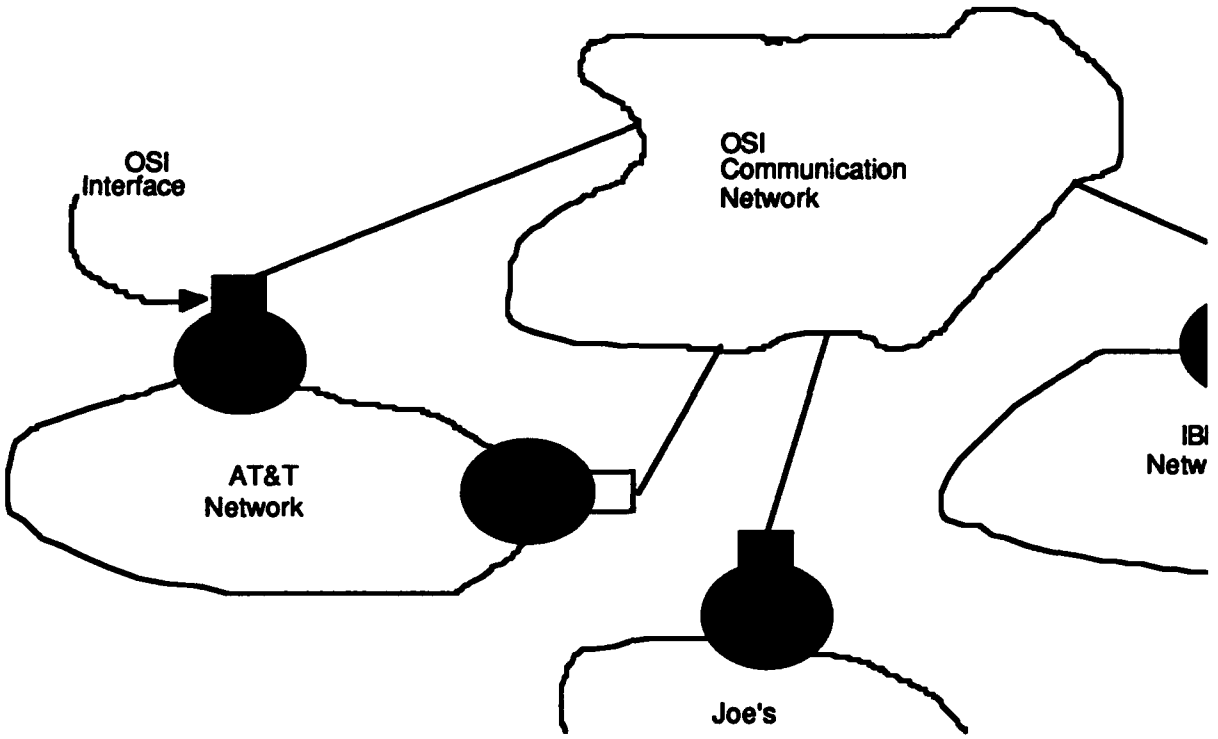


Figure 1-11. OSI communications support for systems management processes

The goal of the communications aspect of the management model is to define an architecture that will allow management processes to exchange and understand management information. As shown in the figure, this exchange can occur between management processes provided by a single vendor or between multiple vendors' management processes. Interactions between management processes are accomplished through the exchange of management directives. The directives are transported using OSI communications protocols. To facilitate these exchanges a set of common management directives is used. The OSI Interface is the point where the messages used by the various management processes are mapped into the common directives, as shown in Figure 1-12.

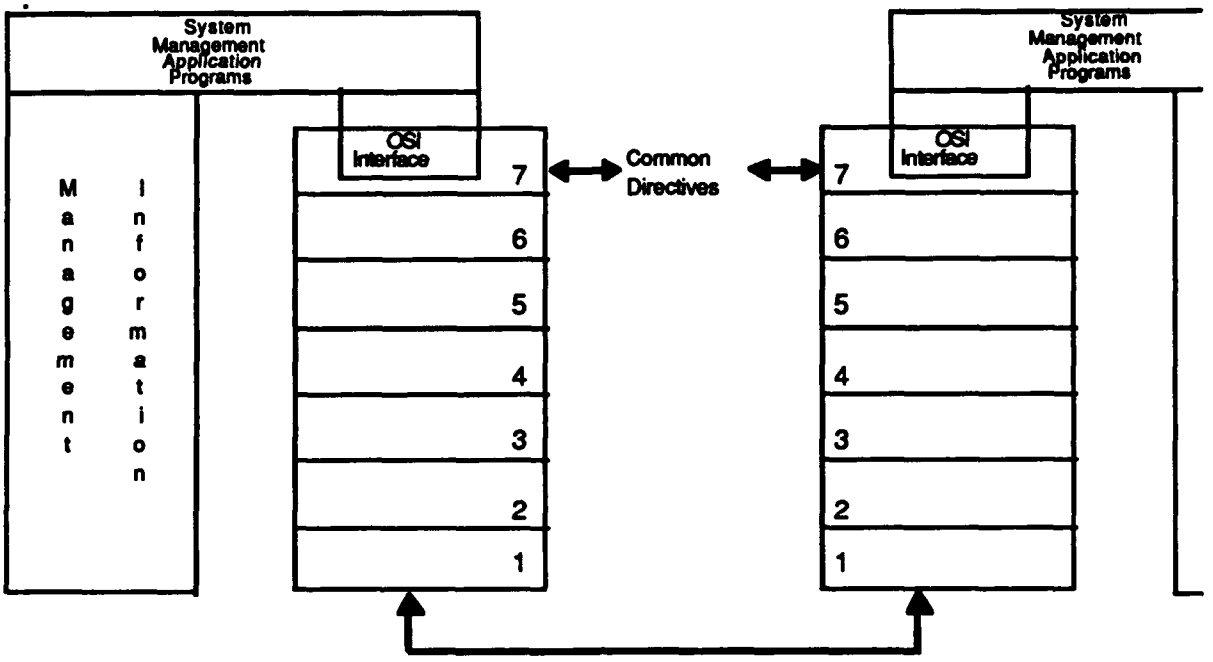


Figure 1-12. The exchange of management directives.

The OSI Interface provides communications services to the application processes. The services provided fall into two categories, management notification and management operation services. Notification services are provided by an Event-Report function. This is a generalized service used to report events, such as alarms, related to managed objects. Management operations include the services required to retrieve information on managed objects, to create managed objects and to request management services from another management application. The OSI processes that provide communications services to the management applications are collectively referred to as a System Management Application Entity, SMAE.

A Systems Management Application Entity is constructed by grouping together combinations of Application Service Elements (ASEs). An ASE is a set of functions used to provide a specific communication service. For example, if you have written a program and you need to access files, such as a remote file server, you would use the File Transfer, Access and Management (FTAM) Application Service Element (ASE). ASEs map service requests from application processes or other ASEs into a set of common primitives used to convey the requests to the destination ASE. The destination ASE will translate the request to an indication which is sent to the receiving application process. The structure of a Systems Management Application Entity is shown in Figure 1-13.

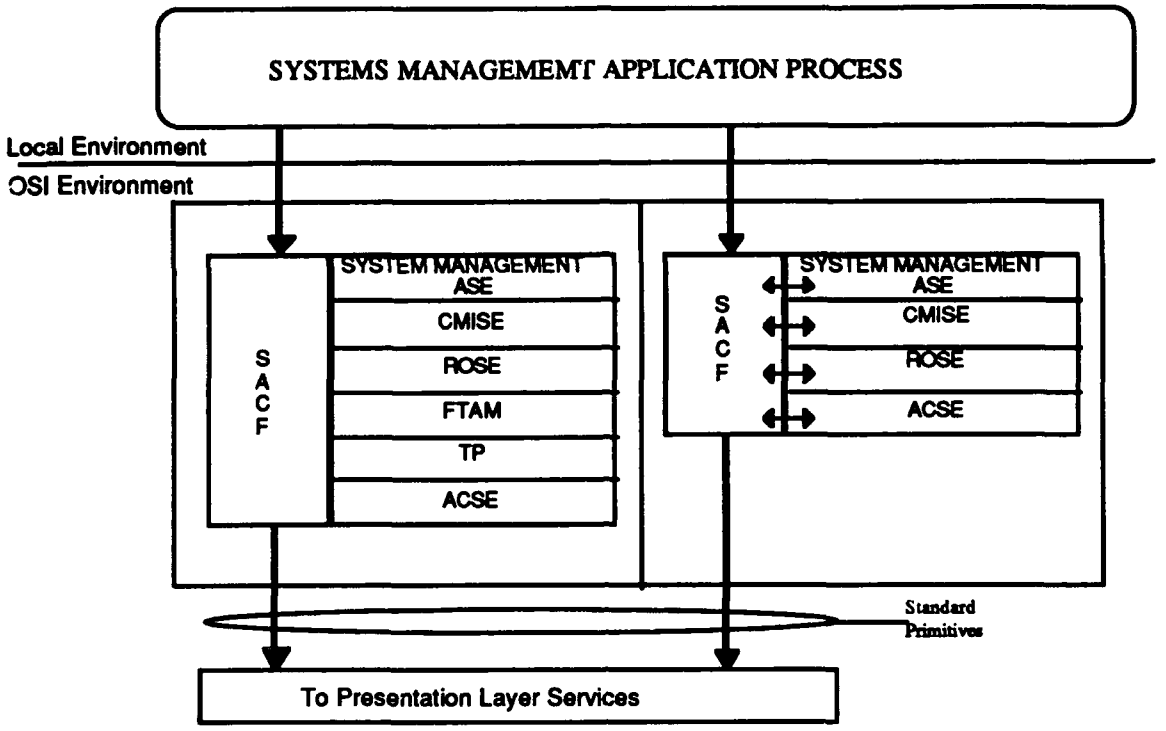


Figure 1-13. Systems management application entity.

The Common Management Information Service Element (CMISE) is the primary systems management ASE. CMISE provides two types of services for the purpose of systems management. The first is the Management-Event-Report service, which is used to report an event about a managed object to a peer CMISE service user. This is the "something happened" service. The second service is used to initiate management operations. This is the "do something" service. Table 1-3 lists the service primitives generated by CMISE.

Table 1-3. CMISE service primitives.

	M - EVENT-REPORT

	M - INITIALIZE
	M - TERMINATE
Management	M - ABORT
Operations	M - GET
	M - SET
	M - ACTION
	M - CREATE
	M - DELETE

To request one of these services the user's application program would send a *Request* primitive to CMISE. For example, a performance measure might require the retrieval of a number that gives the maximum number of connections maintained at a specific SAP. To retrieve this number a performance management message requesting the M - GET service would be sent to CMISE. This request message would contain the information required by CMISE to construct an M - GET primitive to request the specific piece of information. The same M - GET primitive could be used in another instance with different parameters to request a different attribute value from another managed object.

The Remote Operation Service Element (ROSE) supports the ISO equivalent of a remote procedure invocation. This ASE provides services to split a single transaction into multiple requests and to associate multiple responses to the original request. CMISE uses ROSE for its request and responses. The ROSE service primitives are listed in tabled in Table 1-4.

Table 1-4. ROSE service primitives.

RO - INVOKE
RO - RESULT
RO - ERROR
RO - REJECT-U
RO - REJECT-P

Association Control Service Elements (ACSEs) are responsible for insuring that the receiving ASEs exist and that they are capable of engaging in a dialogue. This may require the negotiation of communication parameters. ACSEs must establish an association between application entities before information can be exchanged.

Finally, in Figure 1-13, the Single Association Control Function (SACF) represents the rules used to determine the order in which the ASEs are invoked and how information is exchanged between the ASEs.

The System Management ASE will provide the messages required to perform each of the five specific management functions (Fault, Configuration, Security, Accounting, and Performance). A standard has not yet been defined for this ASE. The following example clarifies the use of ASEs. Suppose you have a System Management Application Process (SMAP) that needs to convey an alarm indication to another SMAP. The originating SMAP would use a function provided by the System Management ASE to communicate this information. The service would be called something like "Alarm-Report". The Alarm-Report function would be included within a library of systems management services. The SMAP would call this function with the necessary arguments to identify the type and source of the alarm. The Alarm-Report function would then use the M-Event-Report primitive provided by CMISE. The idea is to map any number of system management messages into a much reduced set of common primitives. Going back to our example, CMISE would use the ROSE service, RO-Invoke, to convey the alarm message to the destination.

The only ASEs currently required for systems management are: 1) The Common Management Information Service Element (CMISE), 2) The Association Control Service Element (ACSE) and 3) The Remote Operations Service Element (ROSE). The File Transfer, Access and Management (FTAM) ASE is included to help clarify the purpose of an ASE and because it is envisioned that future Systems Management Application Processes will make use of FTAM's services.

The organizational aspect of the model describes the highest level of functionality required in order to perform systems management. Recall that the description of the model's aspects started by presenting the concept of managed objects, then the services available to manage these objects were presented, followed by a discussion of the communication services required to convey management information between management processes. The organizational aspects serve to codify the distributed nature of OSI management.

Two or more management application entities may associate in order to provide a distributed systems management instance. During the course of this interaction the management entities may serve either a managing or an agent role. Each system operating within an OSI environment may contain both managing and agent processes, as shown in Figure 1-14.

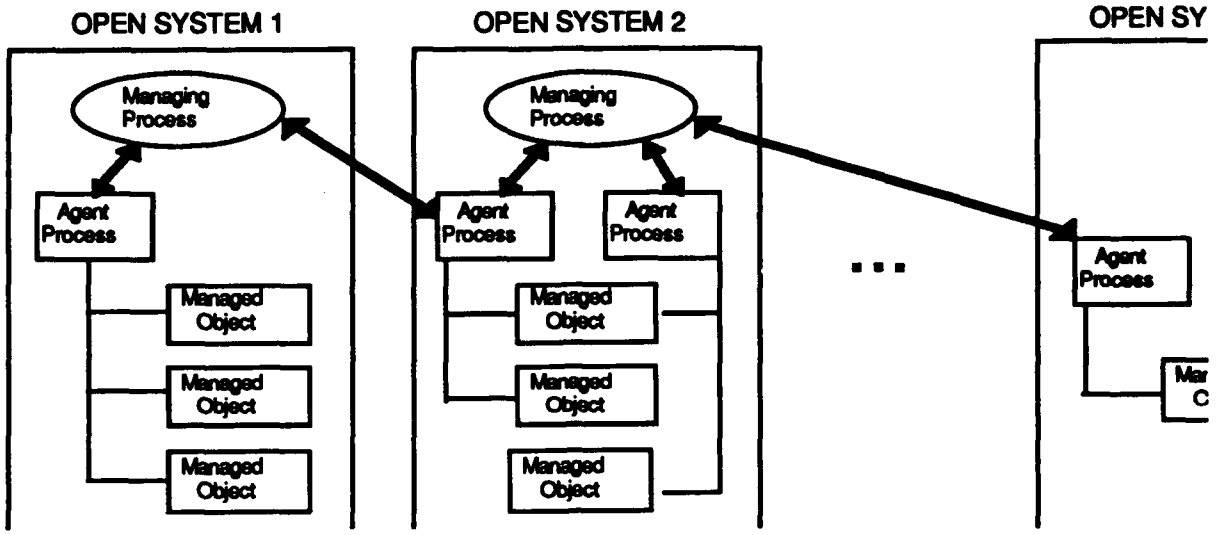


Figure 1-14. Roles of management processes.

The organizational requirements for managing a collection of open systems include:

a) The partitioning of management services into functional groups with an associated policy for each group. These groups are the specific management functional areas of fault, configuration, accounting, security and performance management.

b) The ability to assign and modify the roles of manager and agent.

c) The ability to exercise control, in a consistent manner, over resources owned by more than one open system. An example of this would be the enactment and enforcement of a security policy.

A Management Functional Domain (MFD) is a set of open systems (devices) which has been organized to meet the above requirements. The concept of and interaction between management functional domains is shown in Figure 1-15.

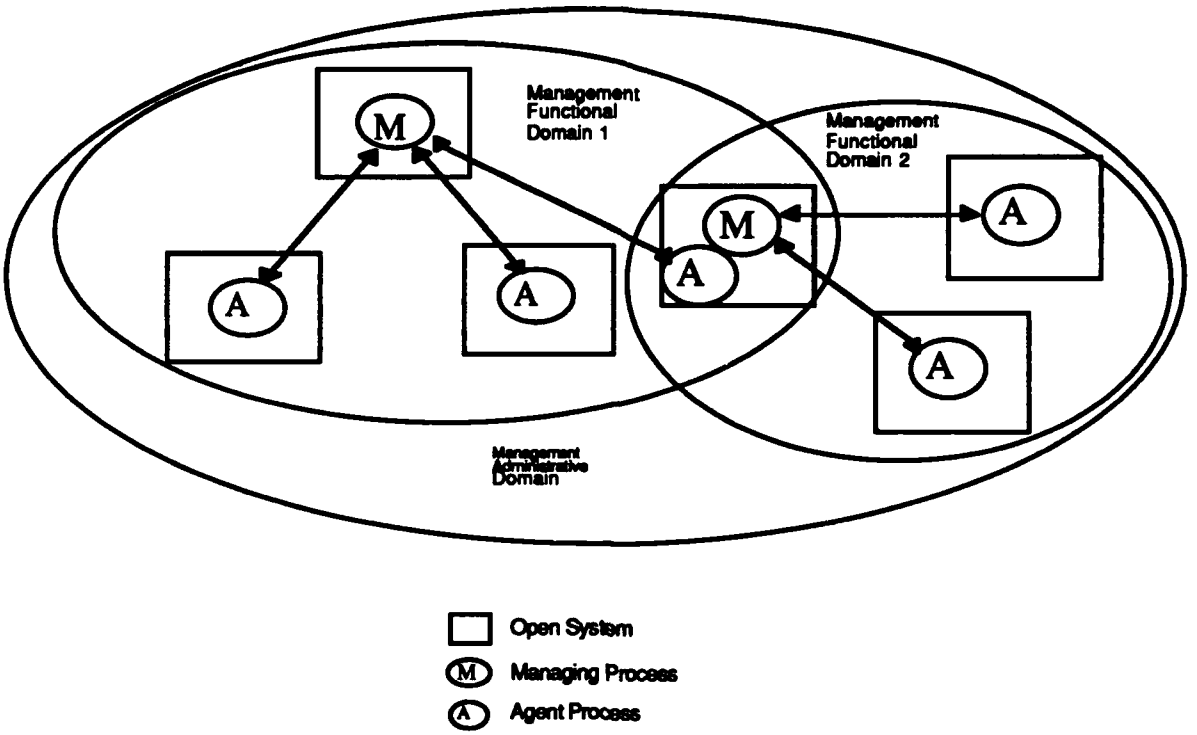


Figure 1-15. Structure of management domains.

As shown in Figure 1-15, sets of management functional domains are organized into a Management Administrative Domain (MAD). This hierarchical organization is like having local system administrators (MFDs) who are all controlled by a group administration policy (MAD). Management administration functions are: 1) the establishment and maintenance of authorities for each MFD, including the ability to modify MFD boundaries, and 2) the assignment of resources (systems) to individual MFDs.

1.3.2.2 OSI Management Standards

The actions which result in the agreement on an ISO International Standard are time consuming. Because its membership consists of standards organizations from 89 different countries, half of the time is spent on technical considerations and the other half on political considerations. The process begins with the production of working papers. These papers are typically written and reviewed by a working group within one of the ISO subcommittees. After agreeing to the overall context of the working paper, the work group writes a Draft Proposal (DP). ISO members have three months to review and comment on the DP. The Draft Proposal is then revised, taking the received comments into consideration. The result is issued as a Draft International Standard (DIS). Members have six months to review the DIS after which a vote is taken. If 75% of the voting members agree to the proposed DIS, it becomes an International Standard (IS). The process may be further delayed by the recommendation of Proposal Draft Addendums. The entire process can easily span a period of four or five years.

For conceptual purposes, the entire spectrum of OSI Management Standards may be viewed as being comprised of four elements. Each of the four elements covers a different aspect of systems management. The four elements are model definition, systems management functions, management information and management protocols.

This set of standards is given by ISO/IEC 7498-4, *OSI: Information Processing Systems - Open Systems Interconnection - OSI Management Framework.*. This paper, an extension to the original OSI Reference Model, introduces the concepts of systems management, layer management and communications protocol management functions.

Currently, this set contains seven items. Each item defines a particular management function, such as an object management function, a confidence and diagnostic testing function and an error reporting and information retrieval function. Some of these items reference the actual messages that are to be employed in communicating the information required to invoke the function.

The management functions are grouped into the Specific Management Functional Areas (SMFAs), such as fault and configuration management. The current focus is to define the required functions and their related messages. Each function and message is assigned as a member of one or more SMFAs.

The four papers in this set describe the organization of information used by OSI management applications. One paper describes the structure of managed objects, including the concept of object attributes and the process used to assign a name to classes of managed objects. A second paper defines a group of object attribute types that may be applicable to most managed objects. Included as part of the attribute definition is the Abstract Syntax Notation (ASN.1) definition of the attribute. This is an ISO standard structure used to encode information related to the attribute. The third paper defines a number of object classes (groupings of managed objects) that may be used as "superior" classes when defining new classes of managed objects. The fourth paper essentially tells the reader how to use the first three items in this set.

There is only one protocol specifically designed to exchange systems management information. This protocol, the Common Management Information Protocol (CMIP), is used by the Common Management Information Service Element (CMISE). CMIP is defined in two standards. ISO 9595 defines the services, i.e., CMIS, used to exchange systems management information while ISO 9596 specifies the actual protocol used to provide these services. The CMIP standard specifies the use of services provided by another protocol, the Remote Operation Service Element (ROSE) protocol.

CMIS/CMIP addresses acknowledged limitations of the Simple Network Management Protocol (SNMP) to communicate status of large numbers of managed objects in a structured, efficient, timely, reliable manner.

In Figure 1-16, each supported protocol is referenced by name and the ISO documentation related to the protocol.

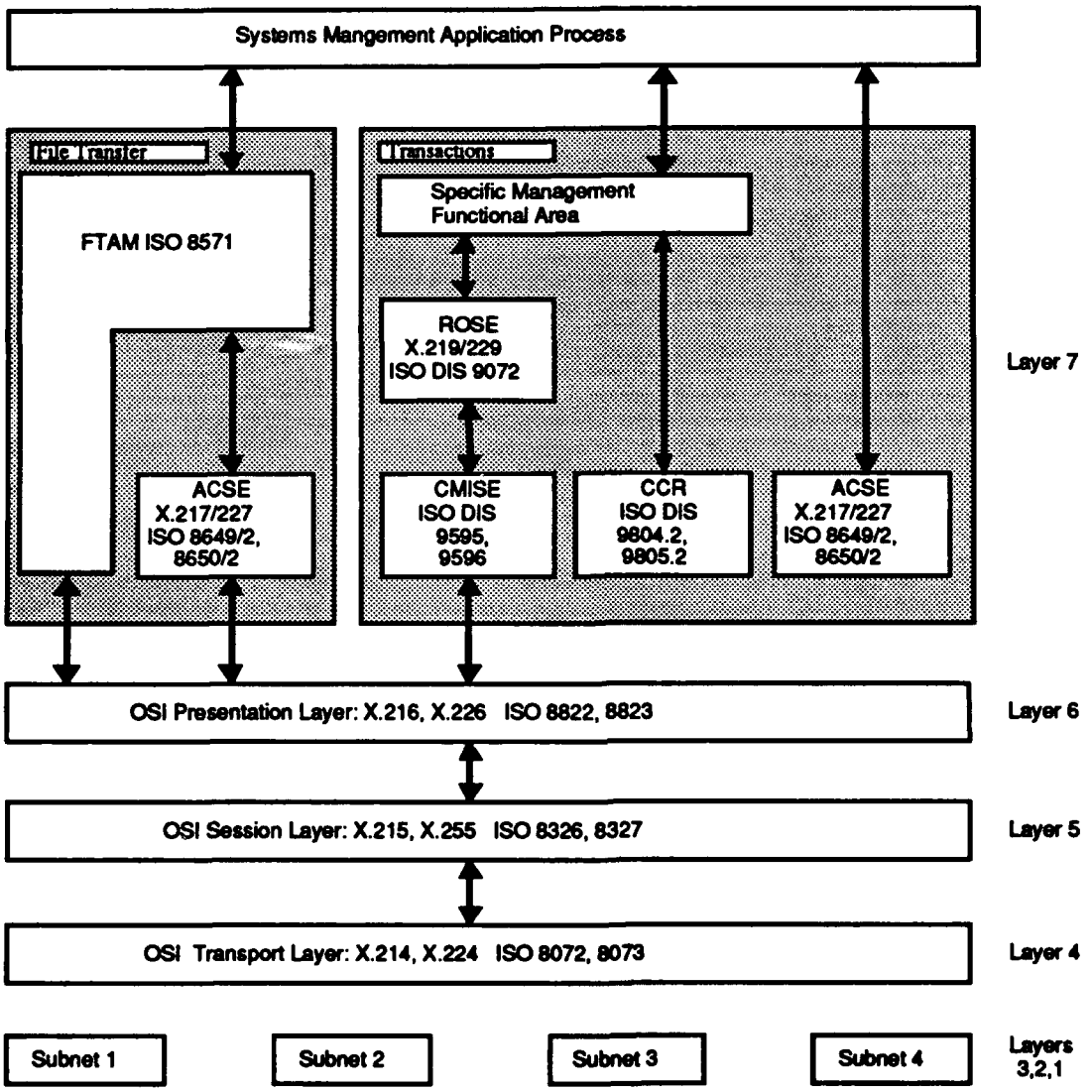


Figure 1-16. Network management protocol stack

1.3.3 Evaluation Criteria

1. Does each system component help the system as a whole to meet the above performance requirements?

2. Is the component "mature?" I.e., is it a non-developmental item?

3. What are the technical, cost, and schedule risks associated with the component?

4. Will the component's performance scale to 10000 entities and beyond?

5. Does the component help the customer on his path to GOSIP?

2. Integration Of Higher Order Models into the DIS/BDS-D Environment

2.1 Introduction

2.1.1 Purpose.

The Integration of Higher Order Models (IHOM) task defines and catalogs the objectives, benefits, and technology challenges associated with the integration of higher order models with distributed simulations such as the Battlefield Distributed Simulation - Developmental (BDS-D). Higher order models are those models and simulations whose operations take place at the unit level, not at the vehicle level. Their scope is much greater in breadth and depth of the battlefield than the current BDS-D environment, but their fidelity of operations and level of time and space resolution is considerably poorer. The IHOM task, conducted as part of the BDS-D architecture study, investigates opportunities to make use of aggregated models, especially of models of echelons above battalion, to expand the capabilities of BDS-D.

2.1.2 Background

There is only one physical world, but there are many ways to perceive it. Each military community perceives the real world according to its unique needs and objectives. For example, the analytical community typically deals with aggregated models of combat to allow many fast running iterations examining weapon system, doctrinal, and force structure trade-offs. Individual vehicles are dealt with primarily in small scenarios and often with a limited number of iterations because of the computer power required to run larger scenarios at the vehicle level. The training community, on the other hand, normally deals with vehicle level simulations except when training commanders and staff at division and higher. Finally, the R&D, engineering, and test communities primarily deal with individual vehicles or weapon systems, but they are generally interested in components or modules of those vehicles and how well they operate under various conditions.

All models are, in some way, aggregated models, i.e. modules are actually aggregations of parts and devices, vehicles are aggregations of modules and components, and units are aggregations of vehicles and other types of platforms. The major distinction is that at the lower level representations, devices, modules, and vehicles are physical elements that are tightly linked. A human (operating himself as a platform made up of human subsystems) can see and touch modules and vehicles on the battlefield. Units are loosely coupled and are more conceptual. Companies, battalions, etc. are physically represented on the real battlefield only through their vehicles, personnel, supplies, command posts, etc. Aggregated models use some level of unit as their smallest element and thus have no direct physical analog on the battlefield. Thus there is a strong rationale for making a distinction between aggregated models and models at the vehicle or

platform level. In Distributed Interactive Simulation (DIS), all units (platoons, fire direction centers, companies, and eventually brigades, divisions, etc.) are represented by observable vehicles or platforms. There is considerable discussion on extending DIS into the aggregated unit world. This report is one of the first steps in addressing that extension.

In addition to aggregating vehicles into units, time, space, man-made effects, and natural effects are also usually aggregated in unit level models. For example, BDS-D models generally send Protocol Data Units (PDUs) several times per second with a maximum update rate of fifteen times per second. Remote Entity Approximation (REA), otherwise known as position extrapolation or dead reckoning, then provides the illusion of continuous motion by filling in between the updates. Faster update rates can be accommodated by DIS, but they are well beyond the threshold of human perception and therefore beyond the general bounds of DIS. On the other hand, it is obvious that a lot of perceptible things can happen in an aggregated model when the length of the time step is from one to ten minutes. For example, individual air defense engagements, MLRS volleys, and Hellfire attacks occur in only a few minutes. In an aggregated model, these individual events would generally be combined with others that occurred during the same time period. Thus, it would be difficult to keep close track on cause and effect.

A similar situation arises in resolving the spatial resolution of units in aggregated models. For example, if units in the model move discretely from resolution cell to resolution cell (and do not interpolate or dead reckon), large "jumps" may be observed if the resolution cells are on the order of three to ten kilometers in width and are a major part of the screen display. On the other hand, units can be portrayed to move continuously even over highly aggregated terrain. In this case, it is only the interaction between the terrain and the unit which suffers. For example, sectors and hexes generally are assigned a given level of roughness, forestation, urbanization, etc. The smaller the resolution cell, the better the resolution. For example, SIMNET operates on 100 meter grid posts. While at one hundred meters considerable detail can be seen in the placement of features and the general orientation of the land, the terrain surface is still simply a straight surface stretched between these posts. As the resolution cells further decrease in size to ten and even one meter, very highly detailed terrain representations can be accommodated. However, the computer power required to upgrade terrain interaction processing from a single one kilometer box, to one hundred ten meter boxes, or even to ten thousand one meter boxes for the same one kilometer area exceeds that available for most battalion level scenarios.

Finally, natural and man-made effects are often simplified in aggregated models to generate approximations "appropriate" to the level of aggregation being portrayed. For example, weather is often played using a simple factor affecting movement speeds and hit probabilities. At the aggregated unit level, more detail only increases precision without increasing accuracy. For example, knowing precipitation rates to a precision of two decimal points is of little value in a model which simply asks if the weather is good or bad. Likewise, a detailed communications or radar propagation model usually provides a representation

with both atmospheric and terrain effects considered in predicting the received power. An aggregated higher order model will generally use a simple "cookie cutter" communications model with a constant probability of reception within the entire area.

2.1.3 Benefits of Interfacing

Since aggregated models have so many problems in representing the battlefield, why would we want to interface BDS-D with them? The answer is that while many basic combat functions such as maneuver are aggregated in higher order models, other Battlefield Operating Systems (BOS) and Operational Operating Systems (OOS) functions such as intelligence are primarily conducted at levels above battalion and either are not or cannot be addressed at the current BDS-D level. Also associated with these higher levels are doctrines and associated data bases for conducting operations which exceed the capabilities of a single battalion to execute, but in which a battalion would participate, e.g. a double envelopment or an exploitation. Also, the models and data bases which generate these BOS and OOS functions are automated in some of the higher level models and could constitute the basis for better representing higher level forces and functions which the battalion is likely to interact with or be affected by. These resources are usually controlled at division, corps, or army level or by another Service. Sensors such as Guardrail and JSTARS and weapons such as ATACMS and BAI could potentially be added to the BDS-D world without the need to generate them from scratch. Likewise, the addition of equivalent higher level enemy capabilities which would observe or engage the battalion as part of the extended battlefield, increases the realism of the planning process occurring at the battalion. The addition of these capabilities would increase the scope of the current BDS-D and the fidelity of the overall battlefield simulation.

Another reason for interfacing or integrating with higher level models is the general direction of DoD policy which directs the development of common or compatible standards across the total regime of military modeling and simulation when used for joint applications. Since there is almost no scenario of interest for larger forces which is not joint, the policy essentially directs all future models to comply with emerging standards. The advantages of interfacing BDS-D with higher order models include not only a better context for the vehicle level BDS-D models (the concern primarily addressed here), but also contributes to greater fidelity for the aggregated unit models by "tuning" them against vehicle level simulations portraying an equivalent scenario. The latter is important because it has traditionally been very difficult to obtain consistent results among higher order models or even realistic results when compared to actual battles, tests, and exercises.

As computer power and communications networking become widely available at reasonable cost, the use of Distributed Interactive Simulation (DIS) standards and common environments will allow more simulations to interoperate seamlessly through common representations at the vehicle level of detail. BDS-D is in the forefront of this movement.

2.2 Assumptions

Based on the requirement of IHOM to expand the scope and capabilities of BDS-D, it was observed that there is little value in interfacing models that are essentially equivalent in capability. This refers to models that operate at the vehicle level, but do not represent higher organizational echelons than BDS-D is already providing in experiments such as the Combat Vehicle Command and Control (CVCC) at Ft. Knox and earlier demonstrations of SIMNET such as WAREX 90-3. Thus, while JANUS, BBS, and CASTFOREM provide fairly realistic representations of the maneuver battlefield and are in widespread use, integrating them with BDS-D would involve considerable effort, would not significantly expand the scope of the BDS-D battlefield or the capabilities of BDS-D, and would not involve the use of the DIS protocols and standards.

It is also assumed that the highest priority for the addition of higher order functions to BDS-D is automated command and control of larger units involving combat, combat support, and combat service support. The second priority would be to acquire automated decision making routines for functions such as the allocation and automated response of higher echelon fire support (including artillery, rockets, attack helicopters, and close air support), sensor allocations, communications, etc. At a lower priority would be routines to automatically conduct logistics support, transportation, maintenance, and similar combat service support functions.

The models addressed are primarily Army models and focus on the ground battle rather than the air or naval environment. Wherever possible, object oriented models with joint capabilities were considered.

Finally, it is assumed that in order to show rapid progress in integration of BDS-D with higher order models, the models recommended must be available for use and experimentation without a significant number of conditions or costs attached. This also implies that an unclassified version is available.

2.3 Approach

The overall approach is shown in Figure 1. It involves creating a methodology to characterize Higher Order Models in terms useful to the IHOM task, collecting information on the models of most interest, assessing the models with respect to their ability to support BDS-D objectives in the short term, identifying taxonomies for integration of models, recommending particular models for specific contributions, and developing an approach to linking at least one HOM with BDS-D. If determined to be feasible, step two experiments would then be proposed to demonstrate that objects and messages generated in the HOM could be passed to the lower level model, translated into DIS formats, and transmitted over the DIS network.

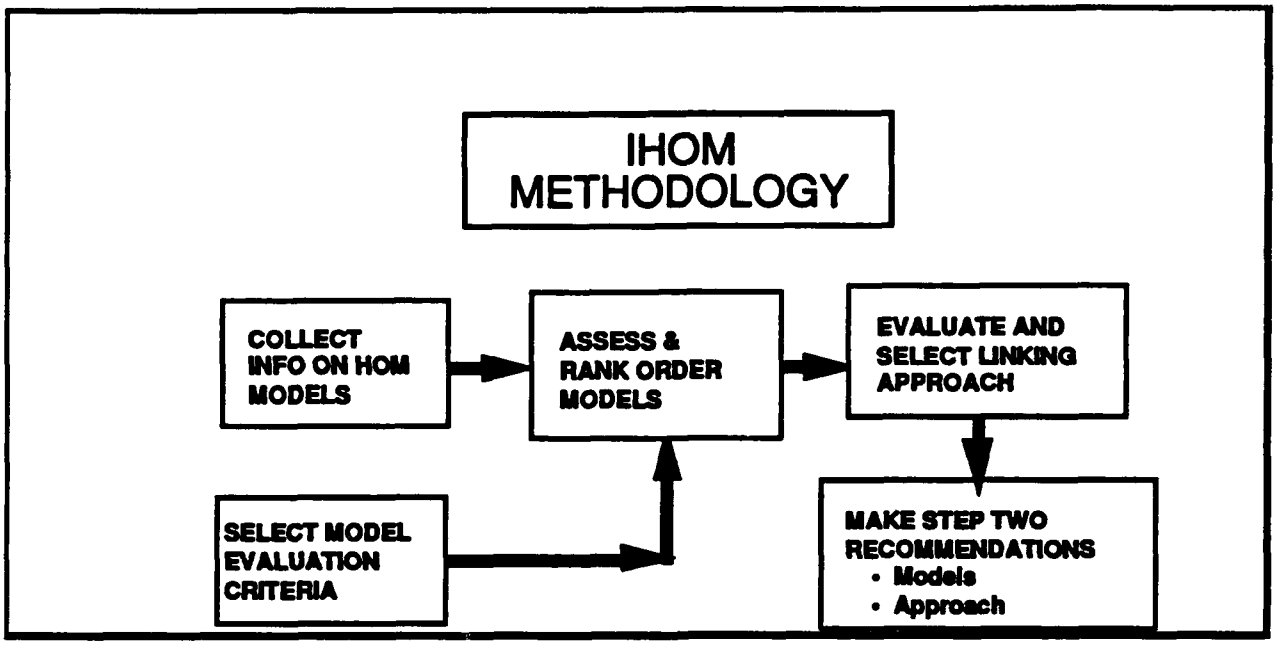


Figure 1. Overview of Methodology for IHOM Step One

2.3.1 Development of the IHOM Evaluation Criteria.

The initial step of the approach identified and described various higher order models and potential evaluation criteria. Step 2 then analyzed and ranked them with respect to their ability to interface with BDS-D. Figure 2 provides an overview of the evaluation criteria selected for this task. The evaluation criteria were selected based on technical, administrative, and cost considerations. Very simple scoring criteria were established to rank order the models. Both the evaluation criteria and the scoring definitions are described below.

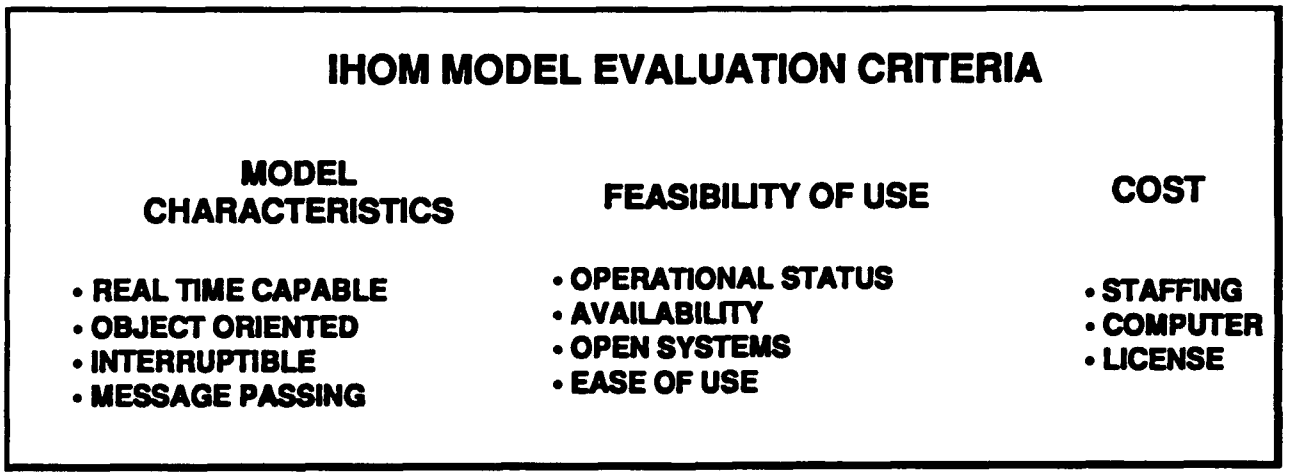


Figure 2. Higher Order Model Evaluation Criteria

2.3.1.1 Model Characteristics.

One of the most basic features of BDS-D is that it operates in real time as perceived by the human participants, i.e. the model must be capable of running at wall clock time. The model must not fall below real time and must monotonically increase when viewed from the outside (internally, the model can be implemented at any speed as long as it results in the perception of real time input and output by the human. The model can therefore include such features as time warping, look ahead, and snap back as long as they are not perceptible to the participants. A scoring of no (0) and yes (1) was given, but if the score was zero, the model was essentially dropped from further consideration. With continuing advances in computer technology, it should be kept in mind that some models which cannot attain real time capability today could do so in the future.

To successfully interface with BDS-D, the higher order models must be object oriented, i.e. they cannot be pure, closed form differential equations in which specific locations of units or vehicles are not maintained. The objects may be aggregated units such as companies, battalions, etc, but they must have a specific location and discrete states which describe the unit's vehicles, supplies, ammunition, etc. The model must also have some reasonable level of fidelity or a mechanism for obtaining that fidelity. The use of ill defined strengths factors, such as Weapon Effectiveness Indices (WEI) or Weapon Unit Values (WUV) is not acceptable for the purpose of this effort. The models considered were all object oriented to some degree with a scoring of no (0) and yes (1). Future evaluations should further examine this area to make better distinctions on degree of object orientation of the selected models.

The higher order model must also be interruptible. This does not mean that the interruptions must be instantaneous, however, the human in the loop must perceive that he can input commands as needed and perceive that his commands are being implemented in a realistic amount of time. These times can be quite large in higher order models since the time between giving an order to move and the unit actually moving can involve hours. This theoretically allows the use of a differential equation model, if the time steps are kept small. However, the use of closed form models will place some restrictions on their interruptability. Most require some minimum time to reach a reasonable solution. In this case, the scoring was divided into three levels, none (0), partial (1), and yes (2).

Finally, the model must have some explicit process of passing messages. It can also have implicit message passing, but there must be a mechanism by which simulation entities (decision making elements) can be isolated from ground truth. The importance of this separation of mental and physical processes (decision makers from ground truth) is fundamental to accurately portraying the fog of war both for the human participants and for the computer generated decision makers. The scoring was simply no (0) and yes (1). Levels of message passing capability and the method of separating mental and physical processes should be considered in future evaluations.

Man-in-the-loop (MITL) was not considered critical in the evaluation of higher order models to potentially link with BDS-D, since only the functions of the higher order model were being evaluated. However, if the higher order model is to be used as part of an integrated training device, then future evaluations should consider the HOM's capability to allow realistic participation in the model.

2.3.1.2 Ability to Use the Model. (Feasibility of Use).

The ability to use a model is dependent on a combination of factors including operational status, availability, openness, and ease of use. Feasibility as used here is as much a political question as it is a technical or administrative question. First, the current status of the model is highly important. Since promises of model availability tend to be exaggerated, it must first be determined if the model is operational (has actually been used), then, since the half life of most models tends to be short, it must be further determined whether the model is still operational. Secondly, it should be determined whether the model has Government sponsorship and whether any validation has been performed. Even if a model is operational, the availability of the model may still be in question since the associated data bases may be classified and therefore restricted or there may be other limitations on release. For the purposes of this evaluation, availability was essentially a go/no-go criteria and was not scored. An indication was made as to whether the model was available directly from the Government or required contractor support. However, the cost and difficulty of obtaining access and porting a model to a new environment could also be considered an availability factor. In this case, there would be major differences among the models.

Also, while not a major restriction in research programs, the "openness" of the model has to be considered. The use of proprietary hardware and software did not preclude consideration of any models for this effort and was not scored. But increasingly, it impacts on decisions concerning which models get Government support and which do not. The evaluation of openness was based primarily on the use of proprietary systems. In this evaluation, UNIX (regardless of which version) was considered an open system. In future evaluations, consideration can be made of the language used (Ada is the Government standard) and whether or not the computer system is Government Open System Interface Protocol (GOSIP) compliant in its communications capability. Note that while Ada is the Government software standard, it is not an object oriented language and techniques for making it object oriented are not standardized, as yet.

Finally, the ease of use of the models is characterized in terms of the type and skill level of manpower required to operate it (including controllers, surrogate players, and operators), the ease or difficulty of the Soldier Machine Interface (SMI), and the general complexity of the model. The last item was not considered a general restriction on the use of a particular model, but could have an impact on cost and schedule due to time required to learn the model or the difficulty in linking with or modifying the model in the timeframe desired. The scoring for ease of use was hard (0), medium (1), and relatively easy (2). A broader scale may be of use to future evaluations.

2.3.1.3 Costs.

Costs were not evaluated in detail, but the amount of manpower required, the size of the associated computer facility, and whether or not a license was required were considered. The basis for characterization of staff size was small, less than six personnel and large, more than twenty. This includes controllers, system maintainers, and other non-player personnel. The basis for scoring of staff size was large (0), medium (1), and small (2).

The computer facility evaluation was based on a scoring of mainframe (0), minicomputer (1), and workstation (2) for the purposes of this relatively small and low cost experiment. None of the models considered required a license and it was dropped as an evaluation criteria. With the emphasis on open systems, it is not expected that it will be a criteria in future studies unless a proprietary system is selected.

2.3.2 Model Evaluations.

A first order evaluation was conducted using sources such as the JCS Wargaming Catalog and the DDR&E(T&E) Handbook of Models. A first order selection was then made based on very basic criteria such as, does this model portray ground combat and is it still operational. This selection yielded a few more than a dozen ground battle models which are in common use and appear to have types of functions not available to BDS-D level (vehicle level) models. Several of the models in widespread use were at the same level of resolution as BDS-D and provided little capability beyond that already represented in BDS-D. These models such as JANUS, BBS, and CASTFOREM are listed below, but they are not further considered as potential higher order models for the purposes of these evaluation.

The second order evaluation considered two groups of models. One group operated only at Echelons of Corps and Below (ECB) and the other depicted operations at theater, which can also be referred to as Echelons Above Corps (EAC). Only one of these models, the Corps Battle Simulation (CBS), is currently in widespread use in the Army training community. However, the higher level Battlefield Operating System (BOS) and Operational Operating System (OOS) functions of interest were found in each of them to varying degrees.

Of the theater level models, both the Joint Theater Level Simulation (JTLS), and Joint Wars along with its component, the Ground Warfare Simulation (GRWSIM) are currently used in the training community. JTLS is in use at the Joint Warfare Center and GRWSIM at the Warrior Preparation Center. Plans are also underway to expand the role of joint level models in new training centers such as Korea. However, as with the corps level models, certain higher level functions appeared to be available within each of several theater level models and five were considered in more detail as candidates to interface with a DIS compliant BDS-D vehicle level simulation.

A short overview of each higher order model is given below followed by an overall ranking of the HOM according to the evaluation and scoring criteria described above. The HOM rank ordering was based on scores for real time capabilities, interruptability, object orientation, message passing, ease of use, staffing requirements, and computer resources. The maximum score possible was eleven and the average score was approximately seven. Since all the models which survived the initial screening are currently in use in the Government, they were all considered to be available. However, it was obvious that for the same cost, some models were much more available than others.

2.3.3 Selection and Ranking of the Models.

The models were initially placed in three generic categories representing the highest echelon of the battlefield which they represented. The first category was models up to brigade. The second category was models with the highest echelon at corps and the third category was echelons above corps. After an initial evaluation, it was determined that BDS-D would not significantly benefit from interfacing it with any of the models in the first category as they basically duplicate the level of operations already available in BDS-D. Category 1 models are provided here for completeness, but they are not evaluated for the purposes of linking them to BDS-D.

2.3.3.1 Category 1: Lower Level Models.

The lower level models considered in this task are the Brigade-Battalion Simulation (BBS), the Battalion Combat Model (BCOM), the Combat Arms Task Force Engagement Model (CASTFOREM), and JANUS. Each of these models is discussed below.

a. Brigade-Battalion Simulation (BBS). The BBS model is currently in use throughout the training community. Its current implementation exercises a brigade or battalion staff along with subordinate commanders. It is implemented at the individual weapon system level and provides a computer generated plan view display of tanks, aircraft, and their engagements with enemy forces. It is manpower intensive to operate, but efforts are underway to increase its level of automation. BBS runs on the Army Family of Simulations (FAMSIM) hardware suite which is basically DEC VAX™ minicomputers and Amiga display devices.

b. Battalion Combat Model (BCOM). BCOM was developed for the Army Research Institute (ARI) using data collected from field tests and exercises conducted at Ft. Hunter-Liggett. The model provides detailed three dimensional displays of vehicle level views. The model can be run in an interactive mode and is currently resident on both SUN™ and Silicon Graphics™ (SGI) workstations. While it portrays human decision making, it has only recently been used in a man in the loop mode of operation.

c. Combat Arms Task Force Engagement Model (CASTFOREM). CASTFOREM is primarily used by the analysis community to examine new

weapon systems or changes to front line combat elements. It has highly detailed terrain and vehicle representations and consequently when portraying large numbers of vehicles, it often operates slower than real time. It is operational on DEC VAX™ equipment, but requires considerable experience to operate.

d. JANUS. JANUS was originally developed by the Lawrence Livermore Laboratories as an analysis tool, but it has been adapted for training and is now in widespread use as a member of the Army FAMSIM. It has a user friendly interface and operates on several types of computers It is most often implemented on DEC VAX™ equipment. It provides a color map based plan view display of individual vehicles with intervisibility considered, but is limited in the size of its battlefield.

e. These and several other models which simulate combat operations at the vehicle level of detail, overlap BDS-D significantly, and consequently have little to offer BDS-D through linking. In the context of this evaluation, they are not considered higher order models. On the other hand, several of them contain algorithms and data bases that should be considered for incorporation into BDS-D as it expands and the models themselves might be used as portals to the BDS-D network. Further investigation of these topics is left for future efforts.

2.3.3.2 Category 2: Echelons Corps and Below.

The models evaluated at echelons corps and below included the Corps Battle Simulation (CBS), the Corps Battle Analyzer (CORBAN), the Combat Sample Generator (CORSAGE), and the EAGLE Corps/Division Analysis Model.

Table 1. Evaluation of Applicable Corps Level Models

Criteria	CBS	CORBAN	CORSAGE	EAGLE
Real Time	Y	Y	Y	Y
Objects	Y	Y	Y	Y
Interrupts	Y	P	P	Y
Messages	N	Y	N	Y
Available	G	C	G	NA
Open	VMS	UNIX	VMS	UNIX
Ease of Use	H	E	M	E
Staff Size	L	S	S	S
Computer	Mini	WS	Mini	WS
Score	5	8	7	11

a. Corps Battle Simulation (CBS). The CBS model is a direct descendent of the Joint Exercise Simulation System (JESS) developed by the Jet Propulsion Laboratory specifically for ground force training at the corps and below. It scored well in basic capabilities, but lost points due to the size of the staff needed to operate it and the difficulty of obtaining such a large staff in a research environment. CBS is particularly manpower intensive when conducting a multi-corps scenario as envisioned by this effort. If a Battle Command Training Program exercise were scheduled using CBS, it could provide the environment necessary to utilize it in experiments with BDS-D. However, there would be few

opportunities to repeat experiments with the CBS model without extensive external support. Even if IHOM experiments were able to "piggy back" on BCTP exercises, the potential would exist for disrupting the training exercises which is an unsatisfactory situation. It should be noted that extensive efforts are underway to link CBS with the USAF AWSIM model as part of the Aggregate Level Simulation Protocol (ALSP) program. Consideration is also being given to linking CBS directly to the SIMNET vehicle level simulators.

b. Corps Battle Analyzer (CORBAN). CORBAN was originally used as a seminar trainer and is currently in use as an analytical model where it is recognized for its ability to realistically represent command and control functions. Its ability to automatically generate more detailed lower level orders based on the model's perception of the battlefield is relatively unique. Only the Eagle model has a similar capability. CORBAN has recently been rehosted in UNIX on a desktop workstation (SUN). While CORBAN is currently operating only in batch mode, its architecture is object oriented and supports real time, interruptible operations. Earlier versions of CORBAN used man-in-the-loop and that capability could easily be reinstated. The CORBAN model has recently implemented on-screen input capability as part of the soldier machine interface.

c. Combat Sample Generator (CORSAGE). The CORSAGE model has been used extensively as an analytical tool to produce scenario data in which to evaluate trade-offs primarily in weapon systems. It has not been used in training and is not fully interruptible by a man in the loop. CORSAGE is relatively easy to use, but requires some experience in its use since it is oriented to the analyst rather than the warfighter.

d. EAGLE. The Eagle Corps/Division Analysis Model will provide a new dimension in corps level combat simulation with highly detailed rather than aggregated representations of close combat and the terrain on which it is conducted. In addition, it will have an extensive command and control structure with explicit communications between all decision making nodes. It is highly automated and can be run by a single operator. It is time stepped, making it most useful as a seminar trainer, but based on its open architecture, it should be modifiable to be fully event interruptible. Eagle is currently in its final developmental stages and should be available for use outside of TRAC in eight to ten months if the current LISP variant is used. In the meantime, efforts are ongoing to directly link it with SIMNET vehicle level simulators as part of an Army proof of principle demonstration.

Based on the evaluation criteria, EAGLE is by far the preferred model to use for the proposed linkage of a corps level model and DIS compliant BDS-D vehicle level simulations and simulators. However, the EAGLE model is under the control of the TRADOC Analysis Command (TRAC) and since it is heavily involved in an effort to interface with SIMNET simulators, there is, as yet, no easy way to access the model for the type of effort envisioned here .

2.3.3.3 Category 3: Theater and Echelons Above Corps Level Models.

Five models at the theater level of scope were evaluated using the same criteria and scoring schema as for the corps level models. These models included the Combat Evaluation Model (CEM) and its relatives such as FORCEM, Joint Wars or the Ground Warfare Simulation (GRWSIM), the Joint Theater Level Simulation (JTLS), the METRIC model of Joint Operations, and VECTOR in Commander (VIC). The Concurrent Theater Level Simulation (CTLS) and the Dynamic Ground Target Simulator (DGTS) were also considered to be candidates for future evaluations.

Table 2. Evaluation of Applicable Theater Level Models

Criteria	CEM	JTWARS	JTLS	METRIC	VIC
Real Time	Y	Y	Y	Y	Y
Objects	N	Y	Y	Y	N
Interrupts	N	Y	Y	Y	P
Messages	N	P	N	Y	N
Available	G	G	G	C	G
Open	VMS	VMS	SIMSCRIPT	UNIX	VMS
Ease of Use	M	M	E	E	E
Staff Size	S	M	S	S	S
Computer	Mini	Mini	Mini	WS	Mini
Score	5	7	8	10	7

a. Combat Evaluation Model (CEM). CEM and its offshoots are essentially large scale differential equation models with little chance of adaptation to a real time, man-in-the-loop, interactive training environment. Its primary advantage is that it has a long history of use and a number of experienced users. However, without an object orientation, it would be difficult to extract functions from the model. Some data bases may still be applicable, but the time available for the follow-on to this effort (approximately six months) precludes further consideration of the model.

b. Joint Wars (GRWSIM). The Ground Warfare Simulation (GRWSIM) is the land component of the Joint Wars model being used and further developed for the Warrior Preparation Center (WPC) and potentially other joint wargaming centers. The model is undergoing modifications to improve its play of several combat support and combat service support functions. As part of the Distributed Wargaming System (DWS), it has also been interfaced to a Deep Attack (Follow On Forces Attack or FOFA) model, an electronic warfare model, and several additional models. GRWSIM is used for training and has most of the attributes needed to provide a highly credible environment for this experiment. However, in its current form, it is manpower intensive for the type and scope of operations envisioned in this effort. Furthermore, it draws several of its BOS and OOS functions such as intelligence from external models.

c. Joint Theater Level Simulation (JTLS). JTLS has been in use in the analytical community for several years. It is object oriented and interruptible, but has been primarily used for corps level scenarios with emphasis on close combat.

Efforts are underway to expand its representation of both rear and deep operations. It is written in SIMSCRIPT™ and has extensive documentation, but has been used more as an analytical model than as a man in the loop training simulation. Several joint level organizations are examining JTLS for applicability as a truly joint level model.

d. METRIC. METRIC is an expansion of the Army Integrated Corps (ICOR) model to theater level with the addition of considerable air and combat service support portrayal. It has been used for both closed form and man in the loop evaluations. It is object oriented and has detailed communications and intelligence representations with explicit message passing. It is a top down model which has global capability allowing for simulation of multiple simultaneous theaters. It also has an adjustable resolution cell (hex) allowing the model to operate at different levels of resolution without changing software.(some data must be changed). METRIC has recently been rehosted in UNIX on a SUN™ workstation. It runs in real time for a full theater scenario including extensive fixed wing and helicopter operations. METRIC is not a Government sponsored model, but has been made available to the Government at several locations.

e. VECTOR in Commander (VIC). VIC represents a multi-corps theater and runs in real time. It is primarily a differential equation model which has been extensively modified to give it many of the characteristics of an object oriented model. It is not generally used for man in the loop evaluations, but could be modified to provide that capability. Because it is not strictly object oriented, any functions extracted from VIC would be difficult to apply to BDS-D.

Based on the evaluation criteria, the METRIC model has a clear advantage over the other theater level models considered. Its strengths lay primarily in its message passing capability and its ease of use. The latter is a function of a relatively simple interface, the ability to run the model with as few as two players (or to repeat wargames with no man in the loop), and the fact that it can be run on a relatively small workstation.

2.4 Examination of Model Linking Taxonomies

Regardless of what models are selected from the evaluation, they will still have to be linked to BDS-D through some mechanism. There are a several of ways to link models and they have met with varying degrees of success. The following section discusses several of these methods and evaluates them with respect to the objectives of integrating higher order models with BDS-D.

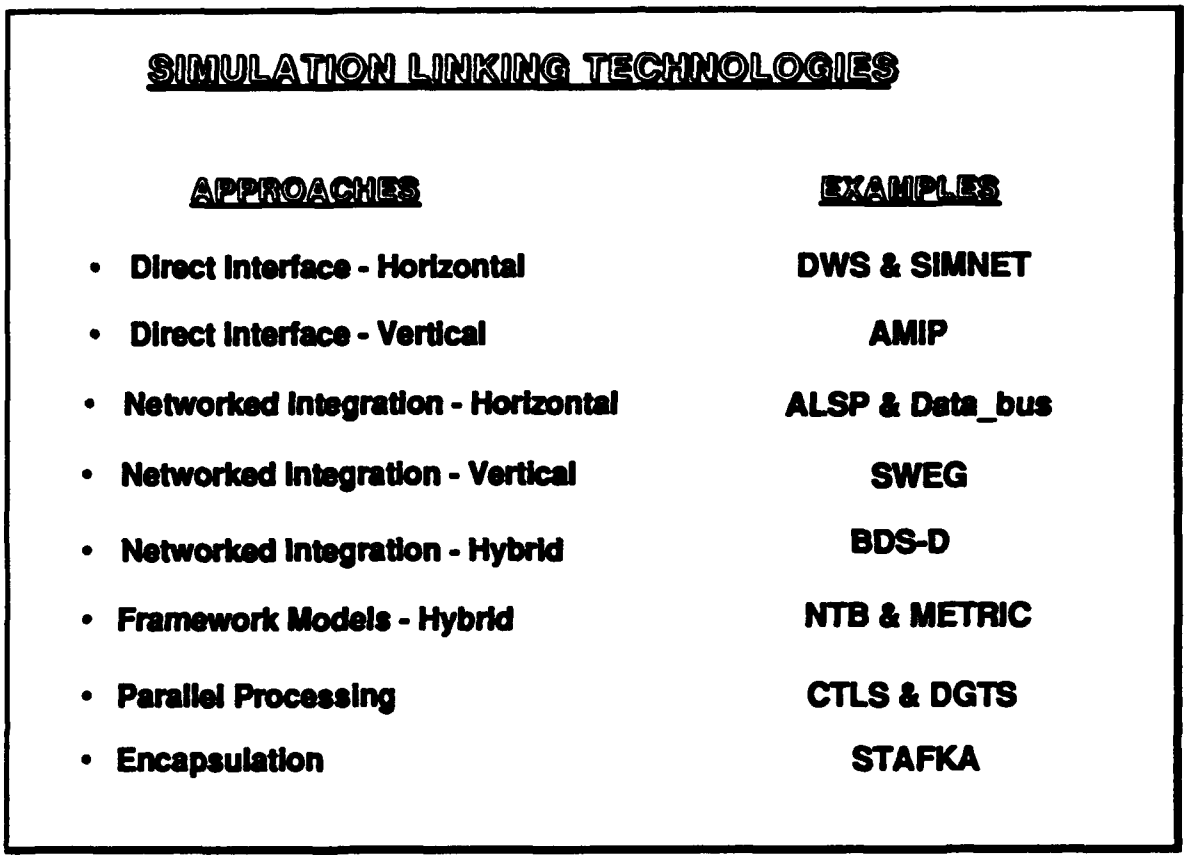


Figure 3. There are Several Simulation Linking Technologies Available. Each has Advantages and Disadvantages.

a. Direct Interface - Horizontal. One of the most popular approaches to linking models and simulations is the direct horizontal linkage of several independent models at approximately the same level of detail. The largest example of this is the Distributed Wargaming System (DWS) at the Warrior Preparation Center where seven separate models have been linked. In this approach, a unique linkage is established between each of the models. However, as the number of models increases, the number of model linkages increases geometrically. The result is large amounts of unique linkage code that has to be maintained in addition to the models. The approach works, but it involves considerable effort and it does not utilize a set of standards to expand the concept.

b. Direct Interface - Vertical. One of the best known approaches to direct vertical linkage of models is the the Army's concept of the Hierarchy of Models. This approach recognized that different echelons conduct different functions on the battlefield and that the command and control structure is hierarchical. However, the Army Model Improvement Program implemented its hierarchy using horizontal slices without standardizing on a vertical component. With separate models implemented independently at each echelon, the command and control function is severed unnaturally. Reassembling a consistent C2 representation is very difficult to do, even when there is general agreement before

building the individual models. If there is no agreement, the task is nearly impossible.

c. Networked Integration - Horizontal. Networked simulation is currently the most popular of the linking technologies and is being applied at several different levels of entity representation. It involves creating standard message protocols and using them to pass needed information among models or simulators at a given level of representation (unit, platform, module, etc.). At the platform level of representation, SIMNET is the best example of networked simulation, i.e. the linking of numerous simulators and simulations (SAFOR).

For models at the unit level of representation, the Aggregate Level Simulation Protocol (ALSP) is being developed by MITRE under DARPA sponsorship to address the shortfalls in the direct horizontal interface approach. ALSP is standardizing the information that must be be passed among equivalent level HOM to link them. It is also building a generic translator for messages containing that information. This is a tough task when the models to be interfaced have common environments, consistent time steps, and a standard data dictionary. However, most current HOM have none of these and each model generally depicts an independent and usually different view of the environment and engagements. Currently, a standard cannot be implemented without making major changes to each of the linked models. Consequently, the ALSP program is significantly modifying both the AWSIM and CBS models in the prototype ALSP implementation. If the functions of the additional models in the Distributed Wargaming System can be integrated into either AWSIM or CBS, then the ALSP standards can probably be generalized. If functions such as intelligence, electronic warfare, and deep attack must continue to be represented by separate models, then it is likely that each existing model will have to be modified significantly before it can be represented on the network. It may be that building a new joint model to the ALSP standards may be easier than making modifications to models that were never designed to be interfaced.

A related horizontal networking methodology for linking HOM is the Data_bus being developed at the Joint Wargaming Center (JWC). In this approach, each model is linked to a common network over which, information is passed in standard formats. This is an inherently efficient way to link models, since there is only one interface for each model. However, the Data_bus does not provide a mechanism or standard for rationalization or normalization among the models on the network. Nor does it have a way of dealing with the large scale redundancy in the models which makes the normalization task so difficult. Thus , it is likely that only new models should be considered for the Data_bus since major changes will be required in older models before they can be interfaced to each other through the network.

d. Networked Integration - Vertical. Messages can also be networked vertically across echelons as well as horizontally at a given level of representation. Usually the implementation is top down. For example, the Simulated Warfare Environment Generator (SWEG) model at the Navy's Air Combat Environment Test and Evaluation Facility (ACETEF) links vehicle and platform level models

with detailed module level models through explicit message passing. Keeping multiple levels coherent is a daunting task, but SWEG handles it by explicitly separating the action or physical world from the cognitive or mental world. SWEG is a product of the test community and is currently applied primarily to air defense environments. It has been used in the past to represent ground combat at the individual vehicle level and it has the advantages of a high level of automation, very efficient software, generation of real time 3D graphics, and direct interface with both man and hardware in the loop. SWEG appears to have solved many of the problems facing the direct vertical interface of a hierarchy of models by separating the physical ground truth from the cognitive operations and dealing with each separately.

e. Networked Integration - Hybrid. The Battlefield Distributed Simulation - Developmental (BDS-D) is an Advanced Technology Transfer Demonstration (ATTD) program to achieve and standardize advances in the technologies first demonstrated in the DARPA SIMNET program. Successor BDS-D programs will set up common environments and link autonomous, heterogeneous simulators together with a relatively small set of DIS message protocols and standard data bases. In current networked simulation, each simulator maintains its own view of the world (time and space coherency is the responsibility of the participating simulator). In future implementations, BDS-D and other related programs will potentially link across echelons allowing multiple levels of representation to operate in parallel. Each model, simulation, or simulator would operate within a common environment using a single ground truth, but each participating entity would carry its own perception of the battlefield appropriate for its echelon.

f. Framework Model - Hybrid. Another approach to linking models is to utilize a hierarchy of models, but to build a message passing capability as an independent model which each echelon in the hierarchy can use. This approach combines the concepts of hierarchy and family of models with an explicit representation of command and control. By using models with the same general architecture and coupling them through their communication systems, a common environment becomes possible. This makes the battleboards (depiction of ground truth) of each model common by agreeing on the basic parameters of the battlefield and a priori determining rules for deciding which model provides the definitive representation of particular entities.

This approach is currently the architecture being pursued at the SDIO National Test Bed (NTB). It is particularly useful if the resulting model is able to make a distinction between ground truth and perceptions and passes separate messages for each (with only the cognitive ones accessible by the decision making elements of the participating entities. The METRIC and ACES models, currently being used by the Army Command and General Staff School and the USAF Wargaming Center, respectively are terrestrial examples of framework models in which more detailed representations of various entities can be embedded.

g. Parallel Processing. There are at least two simulation projects employing parallel processing whose products have potential application to wargaming. The first is the Concurrent Theater Level Simulation (CTLS) being developed by the

Army Concepts Analysis Agency (CAA). The second is the Rome Air Development Center's Dynamic Ground Target Simulator (DGTS). Both models are experiments at producing much higher fidelity simulators than are currently available by employing the computer power available on large scale parallel processing computers. Neither model currently has a man in the loop version which could support this effort.

h. Encapsulation. Encapsulation is a technique for nonintrusive control of an existing simulation. An external framework is used to preprocess all inputs and postprocess all outputs of the model in question. This avoids problems with potentially invalidating a simulation by changing portions of its internal operations or capabilities. Currently, PM TRADE is sponsoring an effort to enhance the BBS model and reduce some of the manpower required to run it by encapsulating BBS with a program called STAFCA. In an unrelated effort, Booz-Allen Hamiliton (BAH) and Coleman Research are currently joined in a effort to encapsulate several models and use the encapsulation process as the translator in a linking approach.

Summary of the Linking Options. The IHOM task does not have the resources and should not attempt to duplicate linking approaches currently being implemented by the ALSP program, the JWC Data_bus, the EAGLE-SIMNET linkage or any of several related programs . It is becoming increasingly obvious that most large scale models at the corps or theater level of operation require almost as much effort to collectively modify as would be required to adopt a new approach with a common environment and an explicit message passing architecture. However, such an approach could not be considered if if it has not been successfully demonstrated. The IHOM challenge, therefore, is to identify existing models which meet the BDS-D criteria for interfacing, i.e. object oriented, message passing, etc; pick at least one such higher order model (preferably a joint model), and interface it in real time with a DIS compliant vehicle level simulation. The best way to do this appears to be a combination of several linking technologies. For example, using a framework model to encapsulate a detailed platform level simulation which has the capability to link upwards.

2.5 Recommedations

2.5.1 General

The functions available from higher order models that would most benefit BDS-D realism involve weapon and sensor systems that could affect the battalion area and the command and control functions that would coordinate these interactions. Higher level weapons include long range artillery such as the Russian D-30, surface to surface missile systems such as the Army Tactical Missile System (ATACMS) and the Russian SCUD, attack helicopters, fixed wing aircraft involved in Battlefield Air Interdiction (BAI) as well as Close Air Support (CAS), and Naval gunfire. Higher level sensors include both Army controlled systems such as the Guardrail and Quicklook aircraft, the USAF JSTARS, and foreign equivalents of these systems.

Based on the evaluation of higher order models, four higher order models appear to have the necessary functions implemented in an object oriented manner, SWEG at the vehicle level of representation, CORBAN and EAGLE at the corps level, and METRIC at the theater level. Of these, SWEG, CORBAN and METRIC operate on a desktop workstation (SUN™) and are immediately available from Government sources. Eagle has an extensive array of additional capabilities, especially in the C3I area, but the model is not scheduled to be available until it has completed testing its interface with SIMNET, approximately the end of 1992.

The IHOM recommendation is to start slowly using the minimum number of models to accomplish the purpose. Therefore it is planned to use the METRIC theater level simulation to create a large scale environment encapsulating the SWEG model (which would be used to approximate a BDS-D model for the purposes of this effort). METRIC would generate both unit objects and specific individual entity objects. These will provide the context for BDS-D including the provision of corps and division weapon and sensor systems. Based on the success of this vertical linkage, the Army model EAGLE or the seminar wargame CORBAN would be examined for additional detailed functions, especially command and control decision making. The METRIC model is a sister model to both SWEG and CORBAN. Since they use essentially the same message passing software architecture, it is anticipated that linking them vertically will not encounter the problems faced by the model hierarchy developed for the Army Model Improvement Program. That is not to say that problems will not be encountered, but the use of a common architecture is a powerful support.

2.6 Proposed Step 2 Delivery Order For Integration of Higher Order Models (HOM) into the BDS-D Environment.

2.7 Background.

In the Step 1 IHOM Task, it was determined that integration of BDS-D with higher order models would provide several benefits. Foremost among these is the command and control context for vehicle level BDS-D battalions and the provision of functions which BDS-D does not currently provide automatically such as higher level fire support and intelligence. Since Army doctrine would not normally commit a battalion to combat without the availability of such assets, it is essential to add them to BDS-D to create a realistic battlefield environment. This Step 2 Delivery Order describes research to demonstrate that the concept of integrating a BDS-D like model with a selected higher order model (HOM) is feasible for a very reasonable level of effort (less than two staff years). It provides an easy and relatively inexpensive path to allow BDS-D to operate within a far more realistic battlefield as it expands its capabilities.

2.8 Approach.

2.8.1 General Overview

The Step 1 Task determined that several object oriented, higher order models (HOM) existed such as METRIC, EAGLE, and CORBAN which could potentially create a compatible combat environment in which BDS-D could be embedded. Furthermore, this could be done without extensive changes to either the aggregated HOM or BDS-D. The initial concept was to run an object oriented higher order model such as the METRIC theater level model along with the the EAGLE or CORBAN corps level model in parallel with a model such as the Simulated Warfare Environment Generator (SWEG) which would represent the vehicle level BDS-D model. After further examination, it was determined that simply interfacing the theater level model would be sufficient if it provided a low enough echelon such as battalion with which to interface.

The models would be loosely coupled by a simulated command and control system for passing orders, delivering sensor messages, and making request for fire support and logistics. Internal to the models and transparent to the participants, special purpose messages would also be sent to tightly couple ground truth in each of the models. BDS-D manned or simulated (computer generated forces) vehicles would simply become more detailed versions of selected unit objects within the higher order model and would assume reporting responsibility for them, both for explicit command and control and for underlying ground truth.

Elements from echelons represented in the higher order model (METRIC), but not in BDS-D, such as resupply convoys, replacements, arriving aircraft, long range artillery, etc would enter the surrogate BDS-D model (SWEG) from the HOM as transition objects for which SWEG then assumes local responsibility for both reporting and ground truth. In this short term effort, all unit objects outside the BDS-D battalion area of interest not already represented as individual platforms would remain in their aggregated form. However, small cells of high fidelity could be created at any point within the higher order model to observe specific vehicle level interactions such as raids. In later efforts, whole battalions could potentially be deaggregated as needed to interface with the vehicle level simulation.

2.1.1 Interactions. Since the purpose of this effort is to provide a larger battlefield context for BDS-D, all interaction between the HOM and the BDS-D model will be at the platform or vehicle level with interacting elements from the HOM being deaggregated as needed to provide the necessary interaction. No direct interaction is planned between aggregated objects and platform level entities. The higher order model would be commanded by a BLUEFOR and an ORANGEFOR commander who receive detailed updates on the battlefield through the higher level model's sensors and communications network. These would be joint commanders with air and land component commanders supporting either in person or as computer generated forces. If desired, a naval component could

be included as the METRIC joint simulation plays all Service resources within a common environment.

The land, air, and naval component commanders would direct their operations using the platform and aggregated level objects available in the theater and corps level simulation, e.g. aircraft, airbases, ships, ports, etc. and interacting with opposing brigades, regiments, divisions, etc. as part of the METRIC simulation. As they enter certain geographic areas, they interact directly with the SWEG model (representing a BDS-D surrogate). These objects would continue to respond to their original orders or flight path, but SWEG would assume responsibility for interactions with both the local forces and the local environment. When vehicles depart the area controlled by SWEG, it passes a message to the HOM scoreboard restoring the original status of the objects and restoring control of them to the HOM. The transition point is basically the boundary at which Distributed Interactive Simulation (DIS) messages start to be generated (ingress) or stop being generated (egress). For objects such as ATACMS or downed aircraft which terminate in the BDS-D area, SWEG would report on the outcome of the engagement and pass messages to its own scoreboard and the scoreboard of the HOM.

2.1.2 Human Interaction. It is anticipated that both the air and naval forces would operate primarily on sets of orders input at the start of the scenario to reduce the complexity of tracking results. Changes in operations would occur automatically as the forces interacted and additional changes could be input to demonstrate that the forces are, in fact, under control and not scripted. The land forces component commander, on the other hand, would have both theater level assets and corps level assets to direct. Using the concept of a semi-autonomous corps operating as part of a joint task force in a theater of operations, a slice of one or more corps could be represented in more detail using a detailed corps level model embedded in the theater level model. Those elements outside the detailed corps are portrayed by the common theater environment down to brigade or regimental level. The corps level model would be played at the battalion level, but displayed to the senior command level at the brigade level to avoid indicating where the additional level of detail was being applied. Within the corps level model, one battalion would be designated to be represented by BDS-D vehicle level Computer Generated Forces. Figure 12.4-1 shows the overall concept of encapsulating smaller models within larger.

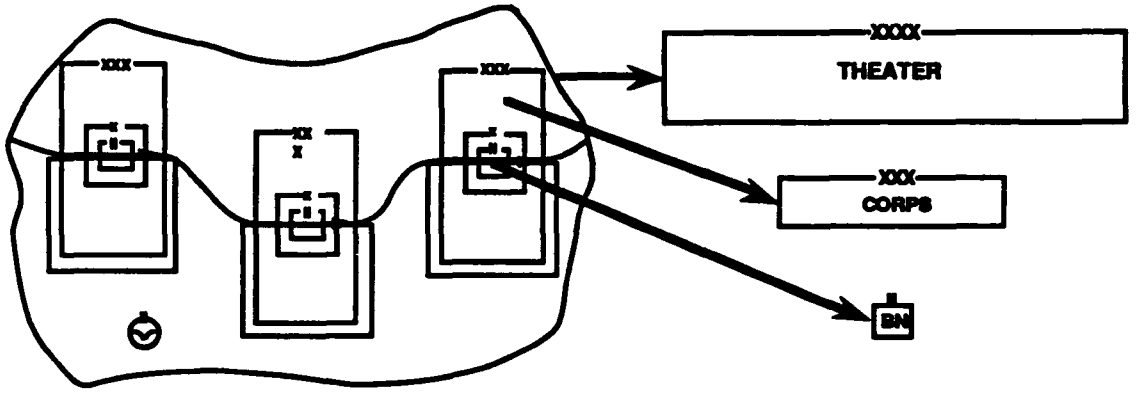


Figure 12.4-1 The Higher Order Models Encapsulate the Lower Level Models

Because the METRIC higher order model being used is object oriented and is not broken into large sectors, it is not tied to FLOT movement (as in TACWAR) or to maintaining strict limitations on one side or the other controlling specific terrain (as in the CBS model). Therefore, the opportunity exists to also conduct relatively small, but highly visible deep operations with airborne or air assault forces. This would be done initially within the context of the theater simulation. However, in future efforts, it could be interfaced with either the existing simulators at Ft. Rucker or with the Rotary Winged Aircraft (RWA) initiative.

2.8.2 Specifics

2.8.2.1 Theater Level. There is presently, only one theater level model in use within the Army which has demonstrated full theater scope, joint operations at the object level, man in the loop capability, and runs in real time. This is the METRIC model in use at the Command and General Staff School. Another feature of the METRIC model is that it operates on a desktop workstation which makes it relatively easy to use.

The METRIC higher order model was recently used to examine Deep Attack options on a multi-corps battlefield and is now being used to examine Army Deep Operations at Echelons Above Corps (EAC). Efforts are also under way to employ METRIC at the Command and General Staff School at Ft. Leavenworth as a prototype model to examine the feasibility of supporting training in joint planning and operations. METRIC uses an architecture that has been implemented in several other models both at the theater level and at more detailed levels down to and including continuous view vehicle level models. Related models at the theater level include FOCUS at the Office of Net Assessment in OSD and the Air Combat Exercise System (ACES) at the USAF Wargaming Center.

2.8.2.2 Corps Level. The concept of integrating the METRIC joint model with a detailed corps level model is very similar to one being discussed at the Army Command and General Staff School. However, at Leavenworth, the student body and the Battle Command Training Program (BCTP) provide large numbers of both warfighters and support personnel and allow the use of the Corps Battle Simulation (CBS) as the corps level model.

Large numbers of support personnel will not be available to conduct corps level operations for this effort. Consequently, CBS is not considered to be a good candidate for this effort. EAGLE scored by far highest in the Step 1 technical evaluation of corps level models, but it is not presently available. Furthermore, the team proposed for this effort has had no opportunity to work with the model. As both of these conditions change, EAGLE will become the candidate of choice for a corps level model to link to BDS-D.

The next highest scorer in the evaluation was the Corps Battle Analyzer (CORBAN). It is presently designated as a Seminar Training tool, but is primarily being used as an analytical tool at Army Concepts Analysis Agency. While CORBAN is not being used as a man in the loop model, it is closely related to CORDIVEM, ICOR, CLEW, and other Army models which have been used with man in the loop. It is currently configured to operate in a batch mode, but an interactive mode is available.

Since CORBAN uses the same architecture as the METRIC model, the current thought is to consider CORBAN components as possible additions to the METRIC model, but to simply go from theater to battalion level and reduce the complexity of the effort by utilizing only two models METRIC and SWEG). Since the focus is on what can be done to enhance the environment and capabilities of the BDS-D level model, the source of the information makes little difference.

2.8.2.3 Battalion Level. For one of the battalions in the corps level simulation and for the enemy regimental units opposing it, the corps level algorithms will be replaced by input from SWEG computer generated forces operating at the vehicle level of resolution on detailed terrain. Updates within the SWEG model will be at a rate and in a format compliant with the Distributed Interactive Simulation (DIS) standards. The interface with the METRIC model or the CORBAN model should be relatively straightforward since the all use a similar data driven approach and a common architecture that inherently separates physical actions and cognitive operations.

When unit sized elements such as battalion or regiments interact within SWEG, it will be done at the vehicle or platform level with updates occurring on the order of several times a second. In keeping with the BDS-D objective of expanding the battlefield, this effort will focus on adding new capabilities rather than portraying large numbers of armored vehicles which has already been demonstrated. SWEG assumes the responsibility for generating the necessary messages to establish and maintain objects received from the HOM on the virtual network. For the purposes of this effort, it is likely that the virtual network will be simulated by shared memory to reduce the complexity of the interface task.

SWEG will also be responsible for maintaining communications with the higher level models for all objects under its control even though the BDS-D battalion will not be directly involved. The higher order model has no responsibility for objects within the SWEG area of control and no direct interaction with any objects under SWEG control. However, the HOM will still have its scoreboard updated regularly or for for events which it recognizes. This allows higher level interactions to occur (such as long range radar tracking) outside of the BDS-D environment.

To keep the problem of seams under control and focus on vertical integration, this Step 2 effort will keep most HOM units separated from the detailed model units at distances that limit direct visual interaction. Results of battles at the platform level will be reported upward in messages along with requests for support. These will pass through the various levels of the hierarchical command and structure as they are represented in the corps and theater level models and will be available at every echelon. At appropriate levels they will be aggregated to more closely approximate actual status reports.

The level of controller and map display associated with the IHOM demonstration is dependent on the success of Task 1 of the BDS-D Step 2 Delivery order which is developing an advanced interface to the SWEG software as part of the Computer Generated Forces task. Given that the demonstration of the IHOM interface is successful, in the next step, manned vehicles would participate in conjunction with the CGF. There will continue to be no direct interaction between SWEG and the higher order models at different levels of aggregation. The focus of this and related efforts is to expand the number and types of vehicle level interactions that can occur on the man in the loop, real time battlefield.

2.8.2.4. Vehicle Level. The tank battalion at the Knox SIMNET -T site and the helicopters simulators at the Ft. Rucker site are the primary candidates for conducting the next version of this effort involving distributed manned simulators. Such an interface would be stretching the state of the art for the level of effort and the timeframe proposed here and is not recommended for inclusion in this first Step 2 IHOM effort. This effort focuses on demonstrating a coherent hierarchical environment in which objects and messages are passed from level to level. At the successful conclusion of this effort, with the confirmation that the transfer is occurring smoothly between levels and the DIS protocol data units are being properly generated and interpreted a follow-on should be conducted to demonstrate manned vehicle in the loop. This assumes that within a year either DIS compliant manned vehicle simulators will be available, or a translator will be available to convert from DIS to SIMNET protocols.

2.9 Summary

There are large numbers of higher order models and simulations with which BDS-D could potentially be interfaced to enhance its capability and promote a rapid expansion of its scope. However, most of these models do not have the attributes which would make this interface a straightforward process. The evaluation described above looked at a subset of higher order models that each had

at least some of the characteristics that satisfied the technical evaluation criteria. It was determined early that models at essentially the same level as BDS-D should not be further considered as they brought few new functions to BDS-D and would be difficult to interface without significant changes. The remaining higher echelon models (above battalion) were grouped into two sets. Those that were echelons corps and below and those that were joint models and included echelons above corps. Each was evaluated separately with the idea of working vertically in two steps, i.e. first to corps and then to theater. This was determined not to be necessary if one joint model could span the hierarchy from theater to battalion. The METRIC model was identified as providing that capability and scored highest of the theater level HOM on the criteria evaluation. It is clear that object oriented models do exist at every level of the command hierarchy, but that they have not been used to create an environment for more detailed object oriented models such as BDS-D. Consequently this study recommends that the Step 2 Delivery Order described above be implemented to demonstrate the feasibility of integrating BDS-D models with higher order models to the benefit of both.

It is anticipated that once the proof of concept of interfacing DIS compliant BDS-D vehicle level models to higher order models has been demonstrated internal to the ADST program, the next step would be a full demonstration involving commanders, staffs, and platform crews at the multiple levels being represented. Since the involvement of senior commanders and the commitment of large numbers of troops and simulators is a major endeavor, the IHOM task focuses on demonstrating the feasibility of integrating SWEG (as the BDS-D surrogate) with higher order models in a near seamless manner rather than generating large numbers of vehicles. Upon successful demonstration of this integration, there is a natural progression to the detailed real time simulation of a total theater.

Thus at some time in the future, it will be possible to represent the entire theater at the vehicle level of resolution. However, in the meantime, higher order models can encapsulate BDS-D platform level models such as SWEG, creating zones of high detail within the low resolution higher order model. At the same time BDS-D benefits from the richness of the battlefield context created by the higher order object oriented model. It is highly recommended that such a demonstration be conducted.

3. Object-Oriented Design of DIS

An argument is presented for designing the DIS system in an object-oriented fashion. Five attributes of the DIS Architecture effort strongly imply that an object-oriented approach to its design and implementation is necessary. These are distribution, heterogeneity, reusability, reconfiguration, and complexity, and their implications for DIS are discussed. The software implementation problems of attempting a fully object-oriented design in Ada are discussed and the use of object-oriented COTS extensions to Ada is recommended. Finally the issues of a distributed object-oriented system are examined, and the efforts of the commercial industry standards group OMG (Object Management Group) to deal with these issues are discussed. Cooperation between the DIS community and the OMG is recommended.

3.1 Overview of the Object Model and its Relevance to DIS

Five attributes of the DIS Architecture effort strongly imply that an object-oriented approach to its design and implementation is necessary:

- **Distribution.** The DIS Architecture calls for a distributed system whose elements are provided by different vendors. The message passing between objects paradigm of object-oriented approaches supports the distributed nature of DIS.

- **Heterogeneity.** The elements of the DIS Architecture are heterogeneous in application and implementation (e.g. manned vehicle simulators, Computer-Generated Forces vehicles and Units, aggregate wargames, operational equipment). It is not feasible to dictate a unifying implementation on the elements. Thus data abstraction and encapsulation are methods by which the heterogeneity of implementation for similar or same functions can be supported.

- **Reusability.** The DIS Architecture calls for the integration of extant systems, some of which may not have been initially designed for the DIS. In order to avoid endless and hence expensive pairwise integration, general interfaces will be needed to interpret between the internal requirements of the system and the DIS Protocols. Object wrappers provide a feasible approach to this problem [Soley, 1990; OMG, 1991b].

- **Reconfiguration.** The DIS architecture calls for easy and rapid reconfiguration of simulation entities to create sessions with new combinations of entities and to create entities with new combinations of behaviors and parts. The object model provides a level of control and management over this process not provided by functional methods.

- **Complexity.** The warfighter-in-the-loop, coupled to the scale and scope of the goals, will drive the DIS system to arbitrary levels of complexity. Object-oriented design and implementation has been developed to deal

with the problems created by modern complex software systems that the functional methods are inadequate to deal with.

It is worth reviewing what is, and what is not, an object-oriented approach. An object-oriented approach has three attributes [Booch, 1991]:

- **Objects.** Uses objects not algorithms (functions) as its fundamental building blocks. Objects are data abstractions with an interface of named operations and a hidden local state.
- **Class Structure.** Objects are instances of classes.
- **Inheritance.** Objects are related to each other via type-of inheritance, they inherit attributes from their superclasses.

If any of these elements are missing, then the approach is not object-oriented. It may, however, be object-based (as is Ada, which lacks multiple inheritance and polymorphism) dealing with abstract data types instead of objects).

The Object Model has seven elements. Four of these are major (if any one of these are missing then the approach is not object-oriented), and three are minor (useful but not essential) (see Figure 1).

Elements of the Object Model

Major Elements	Minor Elements
Abstraction	Typing
Encapsulation	Concurrency
Modularity	Persistence
Hierarchy	

Figure 1: Elements of the Object Model Directly Support Modern Software Engineering Practices. There are seven elements of the object model, four of which are critical if an approach is to be object-oriented and three of which are valuable [Booch, 1991].

3.2 Applying Object Model Elements to DIS

3.2.1 Abstraction

Abstraction concentrates on the essential characteristics of an object that distinguish it from all other kinds of objects. Choice of abstraction is thus dependent on a choice of which characteristics are important, and thus many different levels of abstraction are possible. Abstraction provides clean interfaces between objects, and separates the object concepts from any implementation choices. [Booch, 1991].

The DIS Architecture will contain many instances of objects of the same type (e.g. M1 manned simulators) but which are manufactured by different vendors and thus implemented in possibly very different ways. However, each M1 simulator should behave in a manner consistent with that defined for the M1 object. Additionally, many instances will occur of objects of similar types (e.g. different types of tank such as T-72 and M1), and these types will contain similar behaviors, resulting in the possibility of sharing large quantities of code and design effort between the different objects. Thus abstraction is a mechanism for building the taxonomy of objects with their behaviors in the DIS world, and providing a plan to which the various implementers can adhere.

3.2.2 Encapsulation

Complementary to abstraction is encapsulation. Abstraction concentrates on the externals of the object, encapsulation hides the internal implementation decisions from all other external objects. Encapsulation provides explicit barriers among different abstractions, and implements the concept of data hiding. [Booch, 1991].

The DIS Architecture will contain a vast number of differently implemented systems. Maintainability issues alone demand that modifications to one part of the system not require modifications to the implementation of other parts of the system if at all possible. Encapsulation provides a mechanism for handling separate and independent implementation and development of the interacting systems in the DIS environment.

3.2.3 Modularity

Modularity is the property of a system that has been partitioned into groups of abstractions with well defined interfaces between these groups. These groups and their interfaces provide a mechanism for understanding both the complexity of the program and the real world it models by a divide and conquer approach. As with abstraction, many possible partitioning decisions can be made depending on the viewpoint of the user or designer. [Booch, 1991].

The DIS environment will contain many groups of objects (vehicles) corresponding to Battlefield Functional Areas (BFA), or groups of unit belonging to types of superior unit, for example. Modularity will assist in the design effort associated with partitioning the battlefield domain into manageable chunks.

3.2.4 Hierarchy

This is arguably the most important element of the object model. In complex domains there will be more different abstractions than can be comprehended at one time. Encapsulation will help manage this complexity by hiding the inside view of the abstractions. Modularity will assist by clustering logical groups of object. However, these concepts are insufficient, and in a complex domain the abstractions will form hierarchies. Identification of these hierarchies will greatly

simplify understanding of the domain. The two most important hierarchies in a complex domain are its class structure (type-of) and its aggregation structure (part-of). In addition, it is possible to have association structures, where objects are associated with each other even though they are not parts of each other or types of each other. Furthermore, in complex domains the hierarchies will be multiple-inheritance rather than single inheritance. This means that an abstraction may inherit attributes from more than one class structure. [Booch, 1991].

In DIS for example, a manned M1 simulator will inherit from the manned simulator abstraction as well as from the tank abstraction, while a computer-generated forces tank will inherit from the computer-driven simulator abstraction and the same tank abstraction as the manned M1 simulator.

The full type-of hierarchy for the DIS Architecture will form a taxonomy of objects, and will provide two central services to the DIS Architecture. First, it will provide a template to plan implementation of individual systems, select extant systems for integration, and guide the design of object wrappers around extant systems. Second it will provide data input to a knowledge base for reasoning about the domain. This second service is vital, it provides the core knowledge about the synthetic environment known as DIS to all its participants, and most especially to computer-generated forces decision making code. This permits each participant to make reasonable assumptions about how the other participants reason, and thus behave intelligently without having to explain every last detail about every situation. For example, with the hierarchy available a situation map object (belonging to a computer-generated force) could be asked for the number of tanks in an area, and the type-of hierarchy asked for the number-of-tanks attribute of each echelon of unit, thus providing the military intelligence function of aggregating reports of numbers of vehicles into what size unit is present (see Figure 2).

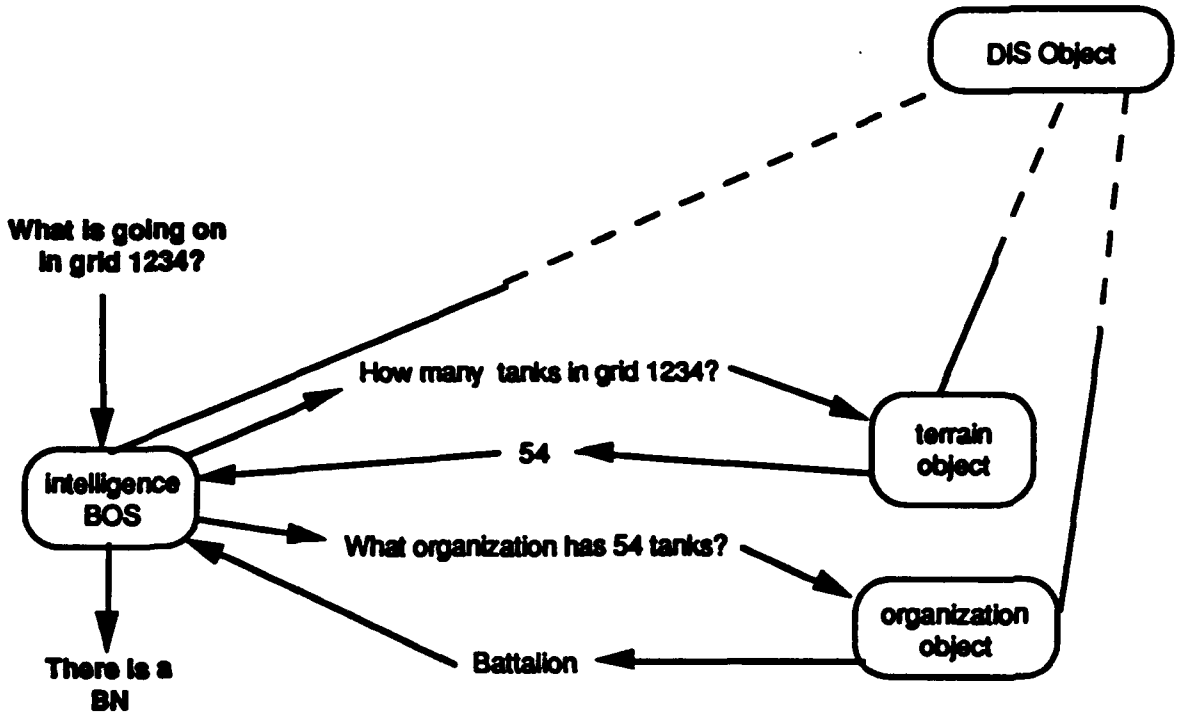


Figure 2: The Object Model Hierarchy as a Knowledge Base. The DIS Object Model Hierarchy is also a knowledge structure available for reasoning by systems within the DIS environment. For example, an intelligence BOS object (belonging to a computer-generated forces unit) could ask a terrain object for the number of tanks in a specific area. The terrain object would pass this back, and the intelligence BOS object could then ask the organization object for command levels consistent with this number. Each of these objects is inherited by specific objects in the DIS simulation system.

A type-of taxonomy of objects is required for DIS. A core taxonomy must be agreed on, and then extended and modified during rapid prototyping during implementation. Not every level in the architecture will necessarily be implemented, some levels and objects will exist in design only in order to facilitate understanding of the DIS system. It should be noted here that the PDU structure mandated in the DIS Protocol Draft Standard [IST, 1991a] is a private matter internal to the network object, and are an implementation decision on the format of messages.

The Object Model of DIS will on occasion disagree with the taxonomy provided by the DIS Protocol Draft Standard [IST, 1991a], and on occasion it will provide additional objects than those found in the Draft Standard. This will provide a mechanism for debating useful changes and extensions to the Draft Standard taxonomy. The DIS hierarchy is proposed as a design tool and methodology rather than a final plan.

Part-of hierarchies are also required for DIS. For example, consider military organizations. An Organization object in a Type-Of hierarchy would specialize to organizations such as Division and Brigade. But we know that Brigades are a **Part-Of** Divisions. Where is this knowledge contained? Each object contains attributes (slots) which help describe the specialization of that object. Organization specializations will inherit from the organization object a "parent unit" slot. These slots will be type restricted, i.e. the "parent unit" slot of any brigade organization will be restricted to objects which are division organizations, those of battalions will be restricted to objects which are brigade organizations, et cetera (see Figure 3). Thus part-Of hierarchies are implicitly contained in the Type-Of hierarchy. However, as part of the architecture we must make all hierarchies explicit in an object-oriented DIS architecture.

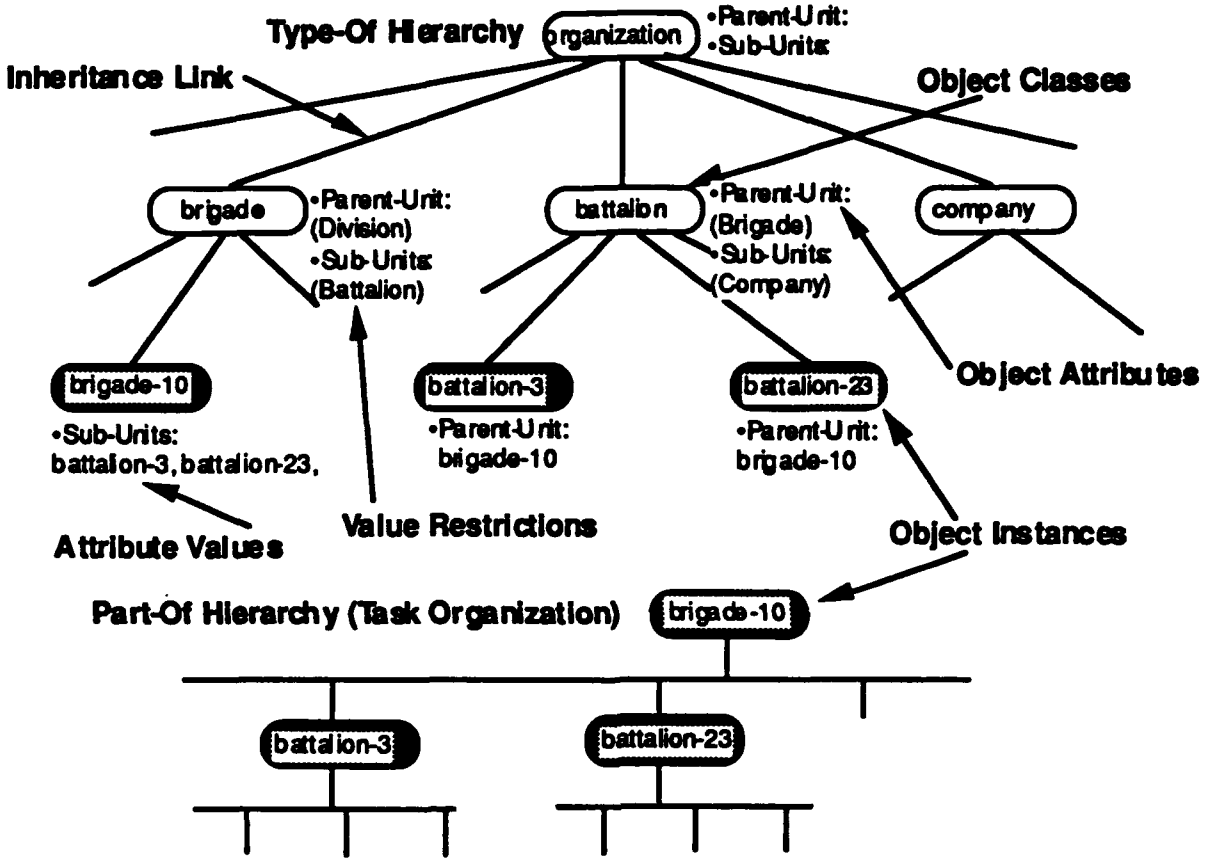


Figure 3: The Aggregation (Part-Of) hierarchy for Military Task Organization is implemented by restrictions (typing) placed on the instance variables (slots) of the unit classes. Other Aggregation hierarchies will exist.

3.2.5 Typing

Typing is a valuable element of the object model, but with the presence of object class in abstraction is not viewed as a necessary element. Type places a different emphasis on the meaning of abstraction. It enforces the class of an object, such that objects of different types may not be interchanged in any but the most restricted ways [Booch, 1991]. However, in specific domains it may be that typing provides overwhelming advantages or disadvantages, and so the decision to enforce typing is orthogonal to the decision concerning the use of an object-oriented approach.

Typing will be invaluable for the DIS Architecture, however, as it will be used to assist in the construction of aggregation (or Part-Of) hierarchies. For example, the class of object called Unit may have instance variables (or attributes) Parent-Unit and Sub-Units. Objects which specialize Unit, i.e. are Type-Of Unit, will place restrictions on the contents of their inherited Parent-Unit and Sub-Units variables by typing them. So, the Brigade Unit class will type its Parent-Unit to be

of type Division, and its Sub-Units to be a set of objects of type Battalion (see Figure 3).

3.2.6 Concurrency

The DIS domain almost by definition calls for concurrent processing in that it is distributed and includes humans in the loop who will generate different events simultaneously. The different events in this case are events that are shared over the DIS network, and are thus public events. Concurrency within an application on the DIS net are private to the implementers of that application. Concurrency is thus a requirement for the DIS Architecture, and for this domain a necessary element in any credible approach.

3.2.7 Persistence

Objects may persist in time on a continuum of existence [Booch, 1991]. Specific instances of manned M1 simulators, for example an instance with given engine reliability, may only exist for the duration of a specific exercise (or until killed). However, certain objects (such as map or scenario objects) may exist from session to session.

3.3 Software Engineering Benefits of OOD

Object-Oriented design and analysis methodologies serve not only the architectural design effort, but the software implementation as well. These benefits naturally accrue because the software is the embodiment of the notional Virtual Battlefield architecture, as well as a key component of the Simulation Support architecture. By tailoring the software implementation to the object-oriented architectural framework, application system builders will reap both cost and time savings. Three features of the object-oriented architecture promote these savings.

- **Superior Modeling Representation.** The object-oriented design facilitates a superior modeling representation of the Virtual Battlefield. Because of the object-oriented focus adopted in the design of the Virtual Battlefield, the conceptual constructs that emerge offer a more orderly and logical way to analyze, describe, document, and model the chaos that characterizes the real battlefield. By following the path of the object-oriented architecture, the software implementation inherits this orderly world view.

- **Software Architecture/System Architecture Correlation.** Because object-oriented analysis and design is a system-building technology, the framework that it defines can house the software implementation as well as the system definition. Essentially, the top-level software structure, its architecture, can comfortably decompose along the same lines of definition and description used for the system architecture.

Thus, engineering effort expended to identify the shape of the BDS-D system can be immediately appropriated into the software design.

- **Reusability.** Structuring the software along the notional lines defined by the object-oriented system architecture leads to enhanced software reusability. Each new BDS-D application starts with a clear description of its specific differences and novelties as a new component of the BDS-D system. This clarity results from the "type-of" class hierarchy used to characterize an object-oriented design. A new implementation can thus be described as a refinement of a class, or a set of classes, already identified and captured in the object-oriented architectural description. Software concepts, if not the actual code, used to define the parent superclasses can be reused in the new application to lend it a canonical structure. By noting the class attribute differences which separate already-defined architectural classes from the requirements of the new application, the application designer gains a clear understanding of the path ahead to successful implementation and integration of his new component.

The combined effect of the above three features equals bottom-line cost and schedule benefits to the software engineering process of system implementation. The following software implementation benefits are gained:

- **Accelerated software Development.** Software development can be accelerated because it can follow the clearly-mapped development path provided by the architectural design. Additionally, concepts and code can be reused from one application to the next, or within the context of the same application as the class descriptions become progressively refined and progressively implemented in software.

- **Reduced Life-Cycle and Maintenance Costs.** Software systems, designed and implemented along the object-oriented paradigm, will feature reduced life-cycle and maintenance costs. Because the implementation software has a more logical and orderly internal composition, because it more intuitively represents battlefield dynamics, one can expect that it will be simpler to design, test, field, and fix. There will be a stronger link between the software organization and what the software models and represents. Bugs will be easier to spot and root out. New personnel will be able to master the software in less time. Natural growth and modification of the software--spurred by continued experience with, and analysis of, the relevant technical challenges--can be more easily accommodated in the software implementation.

A key point to consider is that an object-oriented architectural description, coupled with a software architecture that is expressed along the same lines, gives rise to a new dimension of configuration management (CM) activity, *i.e.* class management. Traditional management of software loads, executable libraries, and software source files is now augmented by management of the class descriptions that emanate from the architectural design.

This new direction of management activities should not be considered as merely an additional burden on the CM function. Class management offers something different from the traditional CM activities.

First, class management offers enhanced potential for software reusability. The notional and software architecture are defined in terms of "type-of" class decomposition. As software development proceeds, it is natural to form the new, lower-level, more-specific classes, by reusing the software implementation of the ancestor superclasses and applying the new and changed attributes to the new descriptions. Given this development environment, management of the class descriptions provides the jumping off point for incremental growth and modification of the system.

Second, unlike object libraries and source files, which constitute "black" system building blocks, offering no insight into the structure of the system without the investment of costly analysis, the class descriptions are open, intuitive expressions of either the notional models or the system building blocks. Proper management of these descriptions encourages review of the evolving system.

Third, object management facilitates cross-development, BDS-D-wide CM. The overall architecture imposes a common structure on architecturally-compliant software through the specification of classes, their ancestry, and their attributes. Software components can be defined and tracked not only through their external-system assigned functionality and names, but also through their implementation of the architecturally-defined classes. System-wide versions can be identified based upon the class definitions. External BDS-D software applications will inherit their revision level based upon their linkage to the set of defined version-related classes.

3.4 Implementation Considerations

3.4.1 Requirement for Object-Oriented Ada

The DIS architecture must take account of the requirement to implement in Ada. Unfortunately, Ada is Object-Based rather than object-oriented, since it is missing some of the essential features of the object model [Booch, 1990; Pascoe, 1986; Stroustrup, 1988]. In particular Ada fails to provide the following [Bach, 1989]:

- direct support for inheritance
- object message passing
- dynamic binding
- instance or class method overriding

The lack of object inheritance seriously impedes the use of Ada in implementing an object-oriented design. At least four approaches are possible in dealing with this:

- Restrict the design to those constructs supported by Ada, and give up the major advantages of object-oriented design in the implementation.
- Design according to the full object model, and optimize to a restricted implementation supported by Ada.
- Design according to the full object model, and wait for object-oriented constructs to be made part of the 2167A standard.
- Use COTS products which provide object-oriented Ada pre-processors, such as the Classic-Ada™ product line.

The design of the DIS Architecture should not be compromised by the current lack of object-oriented Ada, since modifications to Ada will be made to support Object Orientedness later, and since there currently exist COTS systems to provide object-oriented Ada now. The DIS Architecture should therefore be designed to be fully object-oriented.

3.4.2 COTS Standards for Distributed Object-Oriented Systems

3.4.2.1 The Object Request Broker

The distributed nature of the DIS, and the use of heterogeneous and extant systems (which may not have been initially designed as object-oriented systems) calls for a generalization of the object-oriented approach to include applications as objects. The Object Management Group (OMG) Architecture is a possible candidate to provide COTS tools and systems to apply to this problem. The OMG is an Industry Standards Group attempting to devise standards for the development and use of integrated software systems. They believe that the costs and complexities of future developed systems may best be dealt with by using an object-oriented approach. They propose an architecture to provide "interoperability between applications on different machines in heterogeneous distributed environments and seamlessly interconnects multiple object systems" (see Figure 4).

The OMG perceive systems to be objects in their own right, and extant non object-oriented systems are integrated by wrapping them with an object-oriented interface (see Figure 5). A design for the Object Request Broker (ORB) component of the OMG architecture, the message passing facility between heterogeneous systems, has been proposed by two joint teams consisting of DEC/HyperDesk and Sun/HP/NCR/ODI [OMG, 1991a, 1991b]. The OMG architecture contains four major parts:

- "**The Object Request Broker (ORB).** Enables objects to make and receive requests and responses."

- "**Object Services.** A collection of services with object interfaces that provide basic functions for realizing and maintaining objects."
- "**Common Facilities.** A collection of classes that provide general purpose capabilities."
- "**Application Objects.** Specific to particular end-user applications. Non object-oriented extant systems are wrapped by an object-oriented interface to the object request broker."

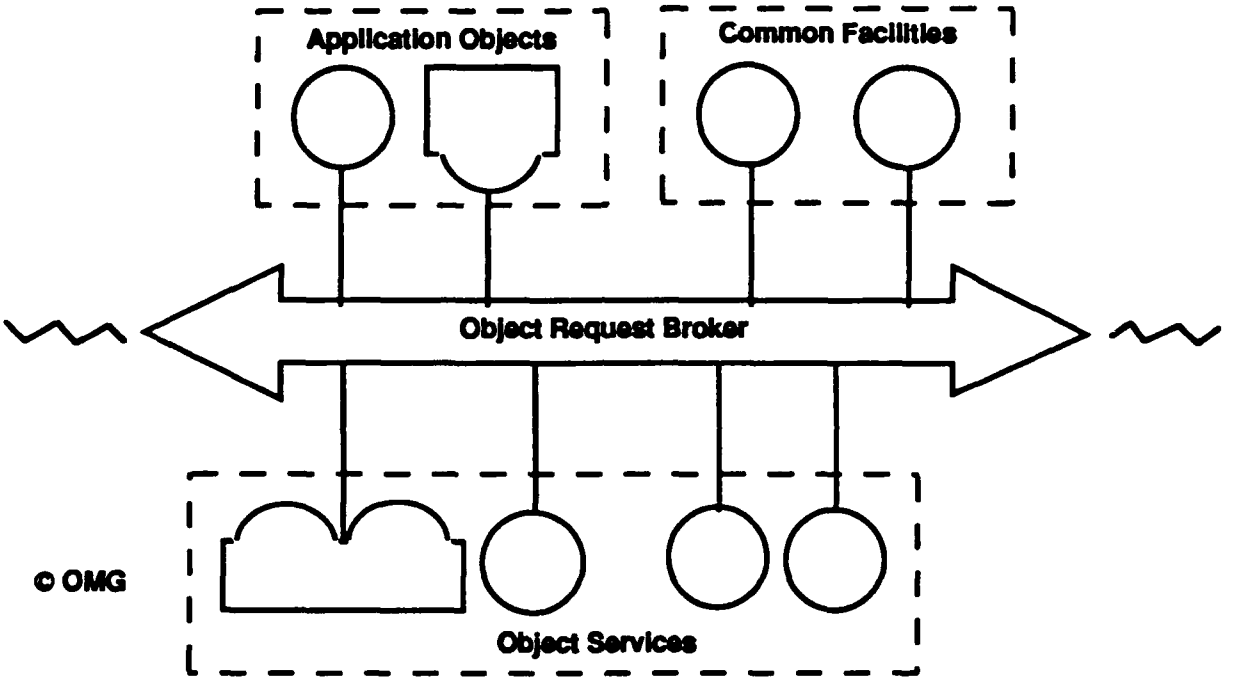


Figure 4: Object Management Group Architecture Overview. The OMG Architecture contains four parts [Soley, 1990]; an Object Request Broker for facilitating communications between objects, Object Services for realizing and maintaining objects, Common Facilities providing general purpose class capabilities, and Application Objects which are particular end-user applications.

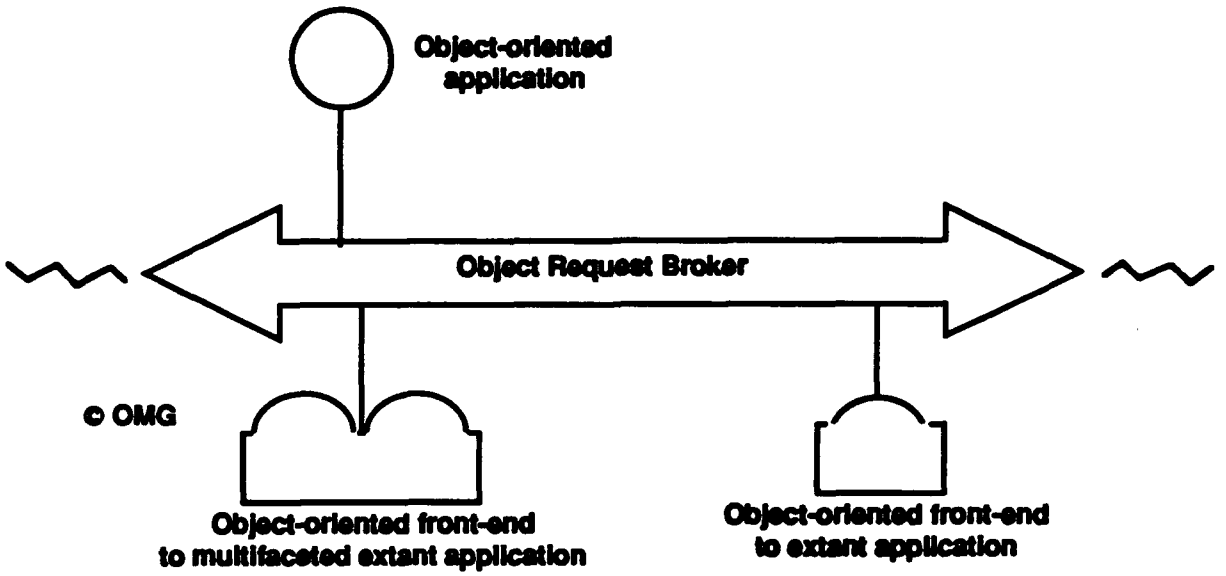


Figure 5: Wrapping Existing Applications. Interfacing general heterogeneous applications within an object-oriented paradigm implies the existence of object-oriented wrappers as interfaces to extant systems which were not built using an object-oriented approach [Soley, 1990; Downes-Martin, 1991].

Part of the OMG effort is aimed at encapsulating, or wrappering, existing (possibly non object-oriented) systems with an object-oriented interface (see Figure 6). In this approach the DIS systems themselves (wargames, operational equipment, computer generated forces et cetera, with their associated hardware) become objects, as well as the simulation objects such as vehicles, regiments, bridges, et cetera. Thus the class hierarchy contains two logical types. The object class hierarchy contains common or global world objects, such as "regiment" for example. Subclasses are specialized classes that describe how these objects are actually treated by specific simulator systems (or operational equipment). The system class hierarchy describes simulator systems (or operational equipment), and captures the exportable behaviors (or messages) and embeds them in a class of message descriptions. One of the class of systems is clearly the user machine interface system, with appropriate subclasses for each system in the DIS system.

The wrapper to each system has two interfaces, the interface to the rest of the virtual reality, and the interface to its own wrapped system. The wrappers are responsible for impedance matching between systems, so that each system observes the virtual reality in its own terms and responds in such a way that global consistency is maintained across systems, simulated objects, simulated (or real) time, and simulated space. In particular the wrappers are responsible for matching the echelon level of objects by expansion and combination. They do so by message passing between the objects that populate the wrapper world. As the DIS system grows, it is clear that an increasing proportion of the overall system knowledge will be contained in the wrapper world, and eventually it will become

easier to extend the system or modify it by direct manipulation of the wrappers rather than the original wrapped systems. Thus the overall system will evolve into a fully object-oriented system over time. Tools for developing wrappers are being developed, and at least one is available as a COTS product (see section 1.5.2.2 [DEC, 1991]), DEC's Application Control Architecture.

The interface between the wrapper and the wrapped system raises an interesting point. If the wrapper is not permitted to intrude deeply into the wrapped system, for political ownership reasons for example, then the wrapper must treat the wrapped system like a black box and restrict itself to manipulating its input and output. This raises temporal coordination issues. One way round this is to allow the wrapper to use the wrapped system to build a sequence of short simulations, rather than a single large one, and to integrate these in the wrapper to provide the appearance of large scale behavior. Then when messages from other objects require manipulation of the wrapped system world, the wrapper can discard the results from the relevant small simulation and create another in the wrapped system using new input data, and then re-integrate this new sub-simulation into the overall simulation.

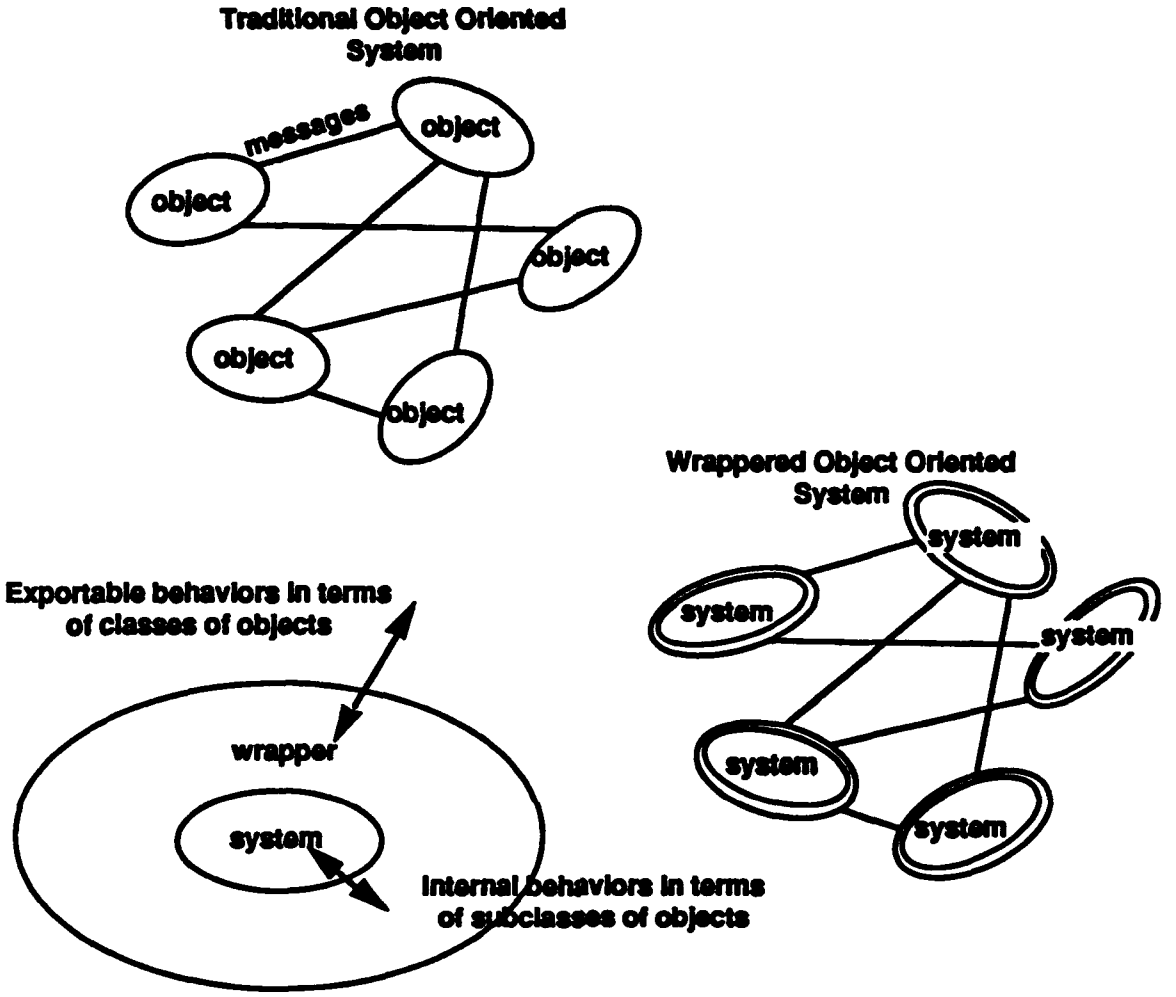


Figure 6: Wrappering, or Encapsulating, a Distributed System. Traditional object-oriented systems consist of hierarchies of objects and classes communicating via messages [Booch 91]. By analogy, a wrappered system encapsulates each system with an object-oriented interface, effectively treating each system as an object [OMG 90. The wrappers encode the exportable behaviors in terms of classes of object and handle the interface to the encapsulated systems in terms of subclasses of object. A class hierarchy of encapsulated systems has to be developed, as well as the more traditional hierarchy of system objects, to handle the exportable behaviors of each wrapped system.

The OMG Architecture provides a clear and compelling set of candidate tools for the DIS Architecture. Although the purpose of the OMG concentrates explicitly on business and civilian applications, the language used is significantly similar to that used to describe Distributed Interactive Simulation. In OMG Architecture terms, DIS systems such as computer-generated forces, wargames, vehicle simulations and operational equipment are simply Application Objects. Extant DIS systems which are not object-oriented would be wrapped in an object-oriented interface. The Object Request Broker would be responsible for the

communications between the DIS Application Objects. Within each DIS Application Object, processing and communications would remain the responsibility of that object.

It is clear that much common ground exists between the OMG and DIS goals. The DIS community should examine the OMG products (both COTS tools/systems and intellectual designs) and determine their application to the DIS Architecture and its implementation. However, the competitive nature of combat introduces temporal issues into DIS not found in the OMG charter. These temporal issues must be separately addressed, and the DIS communities attention is drawn to projects such as the ALSP project [Weatherly et al, 1991; Pullen and Entzminger, 1991] to determine the application of its results to this problem. Nevertheless, certain of the commercial products within the OMG standard are exhibiting message passing rates that lie between 15 and 100 messages a second. These products require analysis.

Application of object-oriented technologies to the DIS Architecture generate a requirement for standards that go far beyond the current efforts for distributed vehicle simulation:

- The creation of global object class hierarchies.
- The creation of system class hierarchies.
- The interaction of wrappers with their wrapped systems.
- The distribution of objects in a time critical environment.

3.4.2.2 Application Control Architecture

A number of business products designed explicitly to assist in generating object-oriented wrappers around extant non object-oriented systems for integration with other systems are being announced [OMG, 1991a], as are other products for implementing the OMG architecture. For example, DEC's Application Control Architecture [DEC, 1991] is an object-oriented software technology that facilitates the dynamic linking of independently developed applications across a network by assisting in the building of the object-oriented wrappers.

"Application Control Architecture (ACA) is an object-oriented software technology that facilitates the dynamic linking of independently developed applications across a network. It does so independently of whether the applications being linked were developed in an object-oriented manner. Different applications can be combined like building blocks to provide unique solutions to business problems, especially in fields such as CASE, CAD, CIM, electronic publishing, and decision support. ACA provides a mechanism for building the object-oriented wrappers around extant applications, and then connects them into the Object Management Group Architecture [Soley, 1990] for integration with other heterogeneous applications [DEC, 1991]."

DEC's ACA technology has been developed in the context of the Object Management Group's Architecture [Soley, 1990] for the commercial business world. However, it is clear that the conceptual similarities between this commercial business related project and DIS are strong. However, the business application world does not appear to deal with the temporal consistency requirements of DIS [Weatherly et al, 1991].

The ACA document [DEC, 1991] discusses the issues facing organizations developing integrated distributed systems today, and how ACA can solve these issues. The document identifies three requirements for integrating existing technologies with new:

- "Existing investments in hardware and software must be supported."
- "Existing and new software applications should be accessible throughout an organization to provide system-to-system interoperability"
- "Existing centralized computing systems at the departmental level should be retained and combined with the advantages of distributed computing environment."

3.4.2.3 The Tool Talk Service

Another recent COTS product of interest to the DIS Architecture is the ToolTalk Service, bundled by SunSoft as part of the Sun operating system. This is a technology to facilitate inter-application operation on distributed networks [SunSoft, 1991a, 1991b]. In addition, much design work and discussion on the technical issues surrounding the OMG charter is available for examination [Powell et al, 1991; SunSoft 1991c, 1991d].

3.4.3 Real Time Performance

Real time performance is a challenge when using object-oriented implementations. The DIS Architecture will provide the full benefits of an object-oriented design and implementation while maintaining real time performance for the simulation. This will be done by clearly dividing the objects into classes that require rapid processing of methods (for example vehicle level objects transmitting PDUs onto the FDDI and reading PDUs from the FDDI which has to happen on the order of many times a second)) and those whose processing time is not rapid (for example Unit Staff objects responsible for mission planning, which occurs over many minutes to hours). Rapid processing methods will be implemented as optimized function calls.

Many of the products which implement the OMG architecture can handle object message rates between ten to a hundred a second. This straddles the internal update rate of the DIS vehicle level objects, is clearly sufficient for PDU transmission rates per DIS vehicle level entity.

3.5 Recommendations

Use Modern Software Engineering. The DIS effort should take advantage of modern software engineering and become explicitly object-oriented. If Ada is mandated, then object-oriented COTS extensions to Ada should be used.

Integrate DoD Seamless Simulation and Industry OMG Architecture. The DoD Seamless Simulation effort should be explicitly integrated with the business Object Management Group Architecture effort to integrate heterogeneous business applications in a seamless environment, and take advantage of the related business products in this area. It is possible for the OMG Architecture to be seriously considered as a candidate paradigm for DIS, and for the work being carried out in the civilian business sector in this area to be exploited by DIS. One approach could be for the University of Central Florida's Institute for Simulation and Training to join the OMG. This would provide a mechanism for inserting DIS requirements into the OMG process, and for the DIS to benefit from civilian business investment in the area.

Appendix A: Computer Generated Forces

Enclosed is a reprint of a Technical Report submitted to PMTRADE under the ADST program.

Open and Extensible Architecture for Computer Generated Forces

Stephen Downes-Martin

February 1992

This white paper is part of Volume II of the Strawman Distributed Interactive Simulation Architecture Description Document ADST/WDL/TR--92-003010.

Prepared by

Loral Systems Company
ADST Program Office
Orlando, Florida

Prepared for

PMTRADE
Program manager - Training Devices
Orlando, Florida

LORAL
Systems Company

Abstract

This Technical Report for Step 1 ADST Task 1 (Advanced Technologies for Computer Generated Forces) includes a summary of the findings of the Step 1 efforts as applied to Computer Generated Forces (CGF) component of the DIS Architecture, describes the problems to be solved, and discussed the technical issues for solving those problems.

The step 1 study determined that four critical operational objectives of the Computer Generated Forces are not being met by the current approaches to the CGF (see Figure 1). These objectives are:

- Provide a modular and open CGF.
- Provide a CGF which incorporates all battlefield functions and is expandable is scale, scope and realism.
- Provide a CGF that supports generation of CGF platforms from the manned platform simulation specification.
- Provide realistic CGF behaviors.
- Provide user interfaces to the CGF which are militarily realistic, do not overload the operator, and which support both warfighters and trainers/analysts as operators.

The step 1 study determined that the CGF operational objectives could best be met by implementing a CGF Architecture that would permit the CGF to be developed, implemented, modified and expanded in independent components by multiple users. This is vital if the BDS-D DIS system is to provide a rich and large scale battlefield at low cost to all DIS participants, most of who will require a simulated battlefield to insert their simulations but who do not have the resources to create such a battlefield. Furthermore, the step 1 study determined that a CGF behavioral approach based on an object-oriented representation of the military C3I structure provided the best architecture for an expandable, modular and open CGF capable of rapid demonstration (see Figure 2). Finally the step 1 study determined that such an architecture is feasible.

Proof of Concept demonstrations of the architecture are recommended that implement the important features of the CGF architecture, demonstrating the feasibility of the modular, object-oriented approach, and showing how independently developed objects can be incorporated to create a CGF which keeps pace with battlefield advances. This proof of concept should demonstrate the following component technologies:

- battlefield operating systems models
- an explicit simulation of tactical communications
- models of general human behavior on the battlefield
- a shared initiative system for alerting the operator when the code is insufficient for the task
- planning tools for senior commanders
- user interfaces

1 Introduction

Computer Generated Forces Study. This Technical Report for Step 2 ADST Task 1 (Advanced Technologies for Computer Generated Forces) includes a summary of the findings of the Step 1 efforts as applied to the Computer Generated Forces (CGF) component of the DIS Architecture, describes the problems to be solved, and discussed the technical issues for solving those problems.

The Loral ADST Team has carried out a Step 1 Study of the Distributed Interactive Simulation (DIS) Architecture. The results of this study are published after extensive briefing to and feedback from the government and industry. Part of the Architecture Study dealt with the role of Computer Generated Forces (CGF) in the DIS battlefield. It was determined by the step 1 study that CGF represented a sufficiently critical and large component of the DIS that the success of DIS depended on a credible architecture for the CGF. Most participants in the DIS effort will want to insert their system into a rich and large scale simulated battlefield, but will not have the resources to create that large scale battlefield (either in terms of providing large numbers of manned platform simulators or computer generated forces). Thus the CGF are an integral part of the DIS infrastructure, and an architecture must be provided for them. This architecture must support open and independent development, modification and expansion of the CGF by multiple users in order to reduce the risks associated with attempting to simulate human decision making on the scale required by a CGF.

CGF Study Scope. The step 1 study for Computer Generated Forces examined four major operational objectives for the CGF:

- **Expansion of the CGF system.** The CGF must grow as the scale of the DIS battlefield grows in order to provide a cost-effective simulated battlefield populated with large numbers and types of platforms. This expansion is in terms scale, realism and scope. This will only be feasible if the CGF can be expanded in a modular fashion, each area of expansion being carried out independently of the others.

- **Scope of CGF behaviors available to DIS.** The CGF must be able to provide a full range of Battlefield Functions and a full combined arms force.

- **Behavioral realism of CGF units.** As the scale of the DIS battlefield increases, CGF forces will represent larger and more complex units than the current company level of command and control. The CGF units and platforms will have to behave with increasing levels of realism and reliability. In particular, CGF must explicitly deal with simulated C3I, and the resultant unit level coordination and synchronization of the CGF units.

- **Use of the CGF by human operators.** As the scale of the DIS battlefield increases, human CGF operators will have larger and more complex CGF under their control. The operator interfaces must be militarily realistic and provide control of the CGF without overloading the operator.

CGF Study Approach. The step 1 study consisted of interview with SIMNET SAFOR staff and students at the Armor School, Fort Knox Close Combat Test Bed, study of reports concerning the use of the SIMNET SAFOR, study of assessments of the SIMNET SAFOR in the scientific press, and analysis by the step 1 study. The step 1 study assumed a CGF is required to provide:

- an open, modular system capable of expansion in scope, scale and realism.
- large numbers of DIS forces with low manpower requirements.
- simulation of the performance and appearance of manned simulators.
- simulation of the performance and appearance of platforms for which no manned simulators exist.
- simulation of human decision making in a simulated C3I hierarchy.
- user interfaces to any decision making node in the simulated C3I hierarchy for the CGF.
- provide credible and realistic large scale battlefield behaviors.

These assumptions, when applied to the study of current CGF technology, lead to the four major conclusions described below.

CGF Study Findings. The Computer Generated Forces is currently treated as a single large system. It precludes independent development of new modules to support specific user requirements. It also precludes exploration of new CGF approaches and implementations by multiple contributors. The CGF however is sufficiently large and complex that it warrants its own architecture with components capable of independent development, modification, and expansion by multiple users. Such an architecture will decouple the risks associated with each technical area of the CGF, and permit a reconfigurable CGF capable of expansion (by multiple independent users) without the major rewrites that have plagued the current system.

The step 1 study determined that there are four critical operational objectives of the Computer Generated Forces (see Figure 1):

- Provide a modular and open CGF

- Provide a CGF which incorporates all battlefield functions and is expandable is scale, scope and realism.
- Provide realistic CGF behaviors
- Provide user interfaces to the CGF which are militarily realistic, do not overload the operator, and which support both warfighters and trainers/analysts as operators

Furthermore, these operational objectives are not being fully met by the current technologies which are used to implement Computer Generated Forces. The step 1 study determined that the CGF operational objectives could best be met by implementing a CGF Architecture that would permit the CGF to be developed, implemented, modified and expanded in independent components by multiple users and application contributors.

The step 1 study examined a variety of techniques that have been employed or proposed for generating Semi-Automated Forces behaviors, as well as some others that might be considered feasible, and an analysis of the benefits and drawbacks of these approaches was made (see Figure 2). The current Combat Instruction Set (CIS) approach of explicitly enumerating all behaviors was determined to provide useful small unit tactical modules. However, it is not expandable to large scale simulated battlefields due to the virtually infinite numbers of behaviors that would have to be enumerated, the lack of a CIS approach to C3I for larger units, and the problems of the rule-based approach to human behavior that the CIS employs.

The step 1 study determined that a CGF behavioral approach based on an object-oriented representation of the military C3I structure provided the best architecture for an expandable, modular and open CGF capable of rapid demonstration. This would also facilitate the transition into the public domain of the current SAFOR CIS modules by embedding them in a simulated C3I structure which supports the large scale battlefield. Thus past investments in SAFOR would be exploited. Finally the step 1 study determined that such an architecture is feasible, and proof of the concept demonstrations are recommended.

Proof of Concept Demonstration for Open CGF Architecture. Proof of Concept demonstrations of object oriented CGF architectural components are recommended in four areas (see columns 2 and 3 of Figure 1):

- Battlefield operating systems models
- General human behavior on the battlefield
- A shared initiative system for alerting the operator when the code is insufficient for the task
- User interfaces

A general object-oriented model of human behavior decomposed into a small number of action and cognition components is recommended, based on the study of alternative approaches to CGF behaviors (see Figure 2). These components are combined into the tasks and subtasks defined by TRADOC's Blueprint of the Battlefield [Pam 11-9], which in turn are combined into Battlefield Operating System objects. The BOS objects are combined into CGF decision making nodes in the CGF command hierarchy. Small amounts of code in defined module libraries may thus be configured in many ways to produce a wide range of behaviors. These behaviors are easily reconfigurable due to the modularity of the approach. CGF objects are distinguished from each other by data which defines the combinations of behaviors they are permitted to perform, thus reducing the risks associated with large bodies of code. The user interfaces are generated by COTS Graphical User Interface (GUI) generators, and determine the access privileges to the various functions of the CGF objects, thus providing both warfighters and trainer/analysts with appropriate interfaces based on a common system. The features and benefits of these technical approaches are described in columns 4 and 5 of Figure 1.

The extant SAFOR CISs (from all government funded SAFORs) should be investigated as CGF platform and small unit drivers to be embedded into the recommended CGF architecture. A robust and mature C3I simulation and wargaming simulation should be identified for implementing CGF C3I and CGF unit behaviors, and behavior combination. One possibility identified by the step 1 study is the Simulated Warfare Environment Generator (SWEG). SWEG is a mature and robust warfare simulation tool for unit and platform levels of interaction used by the U.S. Navy at Patuxent River Naval Air Station (NAS) on the Aircraft Combat Environment Test and Evaluation Facility (ACETEF) and by the U.S. Navy at China Lake on research and development efforts. Both the selected C3I simulation and SAFOR CISs should be used as components of an architecture proof of concept demonstration to test the open nature of the approach and the feasibility of exploiting past investments in CGF and Wargaming.

Technical Innovations. Seven major areas of innovation are recommended as immediate requirements to support the operational objectives of the DIS Computer Generated Forces (CGF) system (see Figure 1). Each of the operational objectives can be implemented independently, by selectively providing the respective technical innovations, thus reducing risk. In addition, each technical innovation has general uses in other areas of defense modeling and simulation, thus providing a clear path for technology transfer to other programs and integration with other programs. The major areas of technical innovation are:

1 **Open and Documented Architecture.** The CGF architecture should be open, with its defined interfaces documented and its code made public. All models used to demonstrate the CGF architecture should be provided as non-proprietary and fully documented. This includes the design and technology as well as the code documentation. Previous investments in SAFOR (using Combat Instruction Sets and other government funded approaches) and Wargaming (using the SWEG Simulated Warfare

Environment Generator for example) should be investigated as modules in the architecture, thus proving the open nature of the architecture. Maximum use should be made of COTS products. The C higher order programming language should be used in order to obtain the benefits of rapid prototyping. The architecture components should be transitioned to Ada on a module by module basis as their utility is proven. This approach will also transition the current CIS based SAFOR into the public domain.

2 **Battlefield Operating System Models.** The design of the DIS CGF should make explicit use of the Army Training and Doctrine Command's Blueprint of the Battlefield [TRADOC Pam 11-9], and similar doctrinal statements from the other services, to build its taxonomy of the objects on the battlefield. The CGF should support complete interchangeability of human operators and software simulations of decision makers via the user interface to the BOS simulations. The code libraries of the BOS simulations should be modular and independent.

3 **Tactical Communications Simulation.** An explicit simulation of the C2 BOS, including Tactical Communications, is recommended to generate realistic unit behavior by providing communications between human and software CGF decision makers. This would provide system hooks for electronic warfare, intelligence analysis, and direct speech communications between CGF and manned platforms.

4 **Realistic CGF Behaviors.** The design and implementation of the BOS based CGF behaviors should be based on a general and flexible human behavioral engineering decomposition of all CGF battlefield behaviors into Action (physical processes) and Cognition (decision making) modules. The Action/Cognition decomposition is sufficiently general that it can be applied to a broad spectrum of battlefield functions.

5 **Alert Agent.** For the forseeable future, code will not be able to completely replace human battlefield decision making. Thus a mechanism is required in the CGF architecture for handling shortfalls in code capabilities. An independent shared initiative system for alerting the CGF operator when code is failing to maintain a desired level of realism should be provided by the CGF architecture. This "Alert Agent" should operate woithout overloading the operator with alerts, and should rank the alerts in order of importance. The Alert Agent module is separate from the behavior generation modules and is independently implemented.

6 **User Interfaces.** The user interfaces to the CGF for warfighters, trainer/analysts and developers should be based on a common system and be distinguished from each other by the level of access granted to different user interface components. Thus a flexible and low cost "dual use" system can be provided, supporting warfighters "fighting to win" as well as trainers, analysts and developers who control the

environment of the warfighters. The user interface windows should be designed and implemented using COTS Graphical User Interface (GUI) Generators.

7 **Behavioral Databases.** Every object in the CGF system, including the definitions of behavior, should be defined by data, and distinguished from similar objects by data. These data objects should be available for inspection, modification and development by non-programmer military experts. By defining the CGF behaviors in an object-oriented fashion using data the CGF architecture should keep the amount of executable code to a minimum, thus reducing risk and maintenance costs and increasing expandability by military experts. These databases are part of the common databases defined by the DIS Architecture.

Once a proof of concept demonstration has been undertaken, an initial architecture with interface specifications will be available to facilitate the specification and open procurement of additional CGF features as resources, budgets, and priorities dictate. The following five technical innovations can be immediately identified as recommended areas of future effort to be planned now:

1 **Transition to Ada** on a module by module basis and provide production level documentation for those modules proven to be effective. Continue R&D on those modules requiring further work.

2 **Integrate the CGF with Higher Order Models** such as the ALSP program (Aggregate Level Simulation Protocol) in which CGF code is used to drive platform level representations of the units being commanded and controlled by the higher order model.

3 **Distribute the BOS components of the CGF** architecture across the DIS network by extending the DIS message protocol to include C3I messages between the CGF components.

4 **Develop decision making tools for the CGF commanders** by expanding on the C2 BOS component of the CGF.

5 **Develop adaptive behaviors within the CGF architecture,** by which the CGF demonstrates automatic learning and exploration using techniques such as neural nets, genetic algorithms, synthetic annealing, and feedback from the alert agent.

Operational Objectives	Technical Challenges	Technical Innovations	Features of the Innovations	Benefits of the Innovations
Provide a modular and reconfigurable Computer Generated Forces system whose components are open to independent development, modification, and validation.	Maintain an open architecture for the Computer Generated Forces.	Object-Oriented Design. Non-proprietary and documented modular models will be used throughout the implementation.	The CGF system is made up small independent modules with well defined interfaces. The CGF modules may be configured in different ways.	Modules may be independently modified or replaced. Development and modification of the CGF is low risk. The military user has control over CGF reconfiguration for future requirements.
Expand Computer Generated Forces to all Battlefield Operating Systems.	Provide an integrated approach to modelling the different battlefield functions.	Battlefield Operating System Models. An object-oriented approach based on TRADOC's Blueprint of the Battlefield (Para 11-9) is used to generate decision making for CGF units and platforms. Realistic Human-Like CGF Behaviors. A general purpose model of human battlefield behavior in terms of object-oriented action and cognition components applicable to all battlefield functions drives CGF behaviors.	CGF decision makers (e.g. command posts) are composed of doctrinally correct BOSs. Action and Cognition behaviors are common to all BOS tasks.	Expandable CGF which is modifiable in the face of changing requirements without requiring major rewrites. Expansion not limited by software. Unit as well as platform level behavior is doctrinally correct.
Generate realistic Computer Generated Forces behaviors.	Generate cooperative organizational level behavior. Use operator intervention to boost software performance without incurring operator overload. Accommodate the full range of battlefield situations without having to write large amounts of code.	Tactical Communications Simulation. Tactical Communications between human operators and CGF software are explicitly simulated as part of the C2 BOS. An Alert Agent Simulation. An Alert Agent is used to warn the CGF commander when the simulation code is faced with a situation it was not designed to handle. Behavioral Databases. Battlefield situations, functions and objects are defined by user modifiable databases used by the BOS models and the human battlefield behavior models.	Human operators communicate with and control CGF objects via the Tactical Communications simulation component of the C2 BOS The alert agent detects the difference between the simulation of tactical failure and any inability of the code to be realistic. The CGF operator can select the thresholds and priorities on which alerts are brought to his attention. The CGF is defined by large databases and small amounts of code.	C3 behaviors are generated, and EW and intelligence is supported. Realistic fog of war is generated. Operator loading is controlled by the alert agent interrupts to CGF operators are minimized to the highest priority and under human operator control. Realistic CGF behaviors are generated at low cost and risk since behaviors are defined in data not code. Military users have complete control over CGF modifications via the database.
Provide a Dual-Use Computer Generated Forces capable of supporting Warfighters "fighting to win" and Trainers/Analysts "controlling the environment" with a user interface based on a common understood military foundation.	Provide a uniform operator interface which supports different user requirements, and provides plug-in capability for operators into any software decision making component of the Computer Generated Force. Provide the ability to create new operator interfaces as required over time.	User Interfaces. The user interfaces provide the operator with access to the C2 BOS of any CGF unit and to the controls of the alert agent. Interfaces with different access privileges provide access to the CGF behavior models and databases. An object-oriented interface to the other CGF elements permits new user interfaces to be created over time. The Alert Agent is used to provide the trainer/analyst with immediate intervention control of the environment.	Human users can control any CGF object via the human interface to the C2 BOS, as the CGF object is controlled by its C2 BOS. The same interface with different access privileges can be used by warfighters or trainers/analysts. The Alert Agent thresholds can be set by the operator.	Only one user interface has to be built for and learned by warfighters, trainer/analysts, and developers. Human operators have complete control over all CGF decisions while maintaining the benefits of automation.

Figure 1 Summary of the CGF Architecture Recommendation demonstrates that the operational objectives for the CGF system are fully supported by the recommended CGF Architecture. Each of the four major operational objectives generate technical challenges which must be dealt with by technical innovations in CGF technology. These technical innovations have features and benefits which not only support the operational objectives, but also provide a technology base for other defense modeling and simulation programs.

2 The CGF Entity Representation

2.1. Comparative Study Overview

Critical and central to the whole CGF issue is that of CGF object representation and behavior generation. Thus this issue is dealt with here in some considerable detail. The CGF must exhibit realistic behaviors at both the platform level, and at multiple organized unit levels of command. When unit level behaviors are considered, the ability to represent human communications, learning from experience, and exploration of new approaches within the CGF is critical if the CGF are to appear realistic. The main technical implication here is the requirement for a representation of battlefield behavior at the general level of human behavior in order to support the simulation of realistic CGF behavior (exhibiting innovation and learning, for example). The representation of human behavior must be flexible enough to support all doctrinal behaviors implied by the expansion objective. The human CGF commander must be able to operate the CGF in such a way that his own military abilities are reflected by the CGF, thus providing a critical element of CGF realism.

A study of the various approaches (implemented and proposed) to generating CGF behaviors was made, and their various advantages and disadvantages considered. A wide range of potential technological approaches are available for use in the CGF. Some of these technologies have been used in current SAFOR systems, and the lessons learned from their use are a valuable indicator of their usefulness. It is clear that a single technology will not suffice, a careful integration of a combination of technologies is required in order to produce an effective CGF. The selection and integration of these technologies is critical to producing a robust CGF system that is low cost, low risk, and satisfies the operational requirements. This must be done without generating a high risk technologically complex integration problem. Four of the various approaches are discussed in some detail here since these are the most commonly used or proposed technologies (see Figure 2):

- **A black box approach**, in which behaviors are modeled by analogy to some physical process.
- **An enumeration approach**, in which behaviors are exhaustively defined in detail.
- **A rule based expert system approach**, in which situations are matched to required behavioral reactions.
- **An object based behavioral decomposition approach**, in which the decision making behaviors of the C3I hierarchy are explicitly modeled on the real world to produce subordinate level behaviors for units and platforms.

The object based behavioral decomposition approach is recommended as providing an architecture that is capable of supporting open and modular development of a CGF which is expandable in scale, scope and realism.

Approach	Advantages	Disadvantages	Conclusions
Black Box	Simple models based on known physical processes.	Human behaviors too complex to model as physics processes.	Cannot represent new behaviors.
Enumeration of Combat Instruction Sets	Simple approach for known behaviors.	Geometric explosion in numbers of CIS behaviors with scale of battlefield. No "tactical control language". Similar to a very crude rule based system.	Cannot scale beyond company. Wrong representation for C3I. Cannot automate combination of CISs into new behaviors
Rule Base Expert System	Rules easily derived from training operations.	Complete set hard to obtain. Rules are poor representation for small unit tactics. Internal structure of rule base is not transparent.	Hard to scale. Behaviors are brittle and break. Hard to modify and update.
Object Based Behavioral Decomposition.	Explicit doctrinal representation at all levels. Behaviors built by combinations of small initial library. Incorporate early investments in CISs. Semantic knowledge base. Automatic behavior modification is possible. Modular open approach.	Potentially compute intensive. Distributed object systems hard to implement.	Unit and platform behaviors are doctrinally correct. Scales to higher echelons without hand coding infinite numbers of CIS behaviors. Can combine behaviors from all services. Independent implementation of components by multiple users.

Figure 2 Analysis of Some Different Approaches to CGF Behavior Generation. Object based behavioral decomposition was judged the best modular long term technology for Computer Generated Forces.

2.2. The Black Box Approach

The black box approach attempts to model the resulting behavior of human decision making in terms of some well understood physical process. A common example is to model platform navigation through a terrain by a potential field theory, in which the platform is given a magnetic (or electric) charge and attempts to maneuver through a terrain consisting of magnetic (or electric) fields. The platform behavior becomes a combination of intention (i.e. an acceleration vector of where the platform is intended to travel) with attractors (valleys) and repulsors (hills, other platforms). The advantages of this approach are, first, that it is based on simple models of well understood physical processes. Second, it is relatively cheap to execute as far as hardware is concerned.

However, the disadvantages are considerable. There is no clear relationship between real world human behavior characteristics and the parameters of the physical parameters. The physical model does not contain a representation of human behavior, thus it cannot be understood by the military user who is not an expert in the physical model. Human behavior and intention have massively many more characteristics than simple physical models. Thus attempts to upgrade the model in terms of changing the simulated human behavior will either require a new physical model that will then ignore the previous behaviors, or will require a massively complex physical model. The latter removes all the advantages of this approach.

2.3. The Enumeration approach

The enumeration approach is that used by the current CIS based SAFOR. It attempts to define all required battlefield behaviors both exhaustively and in detail. The advantages of this approach are that specified behaviors are guaranteed to be present. Any behavior specified by the requirements are explicitly inserted. It is also a very simple approach. The capabilities of the system are exactly defined by the enumeration of behaviors. Finally it provides an invaluable library of vehicle and small unit (platoon) behaviors as input data to a general architecture (but not the final behavioral system, and only if a careful analysis is done of which small numbers of behaviors should be enumerated rather than attempting a complete enumeration).

The disadvantages however remove a pure enumeration model from the list of viable approaches. The number of required behaviors are massive, and a virtually infinite number of behaviors must be defined for a SAFOR that has the same behavioral fidelity as manned units and platforms. This is because there does not exist any approach for automating the combination of behaviors in the enumeration approach, thus every required behavior has to be hand coded. One result of this is that if a specific behavior is not coded, then the system is not capable of reacting with that behavior. Furthermore, simple enumerations of behaviors lead to predictable and robotic reactions. However, most importantly, the approach does not scale to large systems. The enumeration of behaviors in the

current Army SAFOR has reached its effective ceiling at the company level, a massive increase in numbers of CIS will be required for the battalion level if a simple enumeration methodology is followed. This enumeration has proven costly and time consuming to develop.

Interestingly enough, the enumeration approach actually defines a very crude rule-based expert system, in which each enumeration is nothing more than a self contained rule with control over which rules fire embedded in the rules. The step 1 study examined the possibility of adding a battlefield control language to the CIS enumeration approach in order to semi-automate the combination of behaviors into new behaviors (thus providing adaptability and reducing the necessity for hand coding large numbers of behaviors), and concluded that this would produce a conventional rule shell (the CISs being the rules and the control language playing the part of the rule shell itself). This approach would then not only have all the disadvantages of the enumeration approach, but also the disadvantages of the rule based approach to representing human behavior.

2.4. The Rule Based Expert System Approach

The rule based approach is an attempt to reproduce in simple terms the reasoning processes of the decision makers who generate behavior. In its most common form, it attempts to match situations to required behavior and then chain those behaviors together as the situation changes.

The advantages of this approach are that it provides a simple reasoning methodology with well understood programing tools. Each rule is relatively easy to understand by non programmer military users, and rules are relatively easy to obtain from training operations.

However, its disadvantages are considerable when applied to CGF behaviors. Tactical behavior (as opposed to operational or strategic) is better represented by a procedural system with embedded decision nodes, rather than a rule based pattern matching system. Once the situation has been assessed and the mission analyzed, behavior is mainly procedural with initiative being applied to deal with battlefield reality (hence the initial success with the enumeration of CIS approach) via the embedded decision nodes. At operational and strategic levels the situation reverses itself, where behaviors become highly contingent (planning and cognition being emphasized) with embedded procedural segments. A pure rule system is simply the wrong knowledge representation for a CGF architecture. Furthermore, a large rule system is extremely hard to obtain, debug and maintain. Problems here are very similar to those of the enumeration approach. Finally, a large rule system has no explicit thread of control or data through its rules. It thus becomes incomprehensible to non programmers. This effect has been observed in all large rule based systems, such as Digital Equipment Corporation's systems which have resulted in the domain experts being replaced by large programmer teams.

2.5. The Object Oriented Behavioral Decomposition Approach

Object oriented behavioral decomposition is an approach in which the C3I structure of the battlefield is explicitly represented and based on the real world to produce subordinate level behaviors. The advantage of the object based behavioral decomposition is primarily that it provides an architecture that is explicitly based on the objects of the real world and their interactions. As battlefield requirements change, the architecture supports the reconfiguration of the CGF. It explicitly represents the Unit level actions and behaviors as well as Platform actions and behaviors. An explicit representation of doctrine based on TRADOC's Blueprint of the Battlefield [Pam 11-9] is used. This approach avoids the problems of the enumeration approach, in that relatively small numbers of types of object exist, specific objects being distinguished by simple data files. Many objects are represented as combinations of other objects. Each object type is defined and implemented independently, thus reducing risk and increasing code sharing. Similarly, small numbers of simple behaviors are developed and combined to form the full variety of complex behaviors similar to the real world.

The disadvantages that have to be dealt with are that it is potentially compute intensive in execution insofar as it represents the complexity of the real world, and that the complexity of the real world objects may be carried over into a complex model.

2.6. The Step 1 Study Recommendation

An analysis of the advantages and disadvantages of each of the above technologies indicate that no single technology will satisfy the CGF operational requirements. A combination of techniques is required, but this combination must be kept to the smallest set of technologies to manage the complexity of integration. The study team believes that the best CGF is obtained by providing an open architecture based on the Object Based Behavioral Decomposition. The disadvantages of this approach listed above are outweighed by representing much of the behavior and object definitions in data files instead of in explicit code, thus reducing the actual code execution time. In addition, object oriented approaches are explicitly designed to control complexity. Optimization to function calls instead of object message passing at the platform level of execution will ensure efficient use of compute resources.

The study recommends that this approach to CGF behavior generation is demonstrated within the recommended CGF architecture to demonstrate both the architectural concept and the approach to CGF behavior generation. The proof of concept for the object based behavioral decomposition would partition human battlefield behavior into Action (physical processes) and Cognition (decision making) components. The Action and Cognition components would be implemented with a small number of general functions, whose combinations generate an extremely wide range of behaviors. There are only a small number of Action and Cognition tasks (twelve in all), and so implementation at this level becomes low risk. However, by combining these tasks it is possible to generate the

full range of human behaviors once the data that defines each function and echelon have been provided.

3 Open CGF Architecture

3.1. CGF within the DIS Architecture

Figure 3 shows an operational view of the CGF objects and their relationships to each other and to fully manned objects on the DIS battlefield. CGF decision nodes (for example, command posts) are interacting with fully manned command posts, and with CGF platforms and manned platform simulators in mixed task organizations. Note that the CGF contains two top level types of object, CGF platforms and CGF decision makers. The CGF decision makers are abstract, in that they are simulations of human decision making within the C3I heirarchy (along with user interfaces). They are instantiated as the platforms associated with the decision nodes (command posts) along with their internal battlefield operating system objects and intellectual processes. CGF platforms (both combat and decision node platforms) interact with the rest of the DIS world in using the familiar collection of move, shoot, communicate, see, and sustain. The CGF decision nodes interact entirely at the level of information flow, passing orders, requests for information, and information between themselves and fully manned elements via the communications assets within the platforms that are associated with their instantiations. In this architecture all CGF decision making is thus vulnerable to combat damage, both physical (the CGF decision makers can be killed) and informational (the communications between the decision makers and the rest of the battlefield can be disrupted or monitored).

Figure 4 shows the top level DIS Architecture developed by the step 1 study, and indicates the position of the CGF within that architecture. In order to achieve the fundamental DIS goal of interoperation of heterogeneous simulations, it is useful to view the DIS Architecture as a collection of cells interconnected by inter-cell networks. Standard DIS cells are collections of homogeneous simulation nodes, using the DIS message protocol for internal communication. Within a Standard DIS Cell, all simulation nodes use a fully compatible set of simulation models and algorithms; all share a common environment defined by an environment database; and all communicate via broadcast datagram messages. Non-Standard DIS Cells are systems connected to the DIS inter-cell network but whose internal structure is outside the scope of the DIS architecture. In other words, non-standard DIS cells are systems whose internals do not use the DIS standards. Cells themselves (both standard and non-standard) are heterogeneous with respect to each other, and thus their interconnection achieves the DIS goal of heterogeneity.

Multiple CGF entities within a standard DIS cell interact with manned vehicle simulators within the same cell. Additional standard DIS cells are on the network (as are non-standard cells). Examples of manned platform simulators include DIS platforms such as RWA at Fort Rucker or SIMNET platforms such as those at Fort Knox.

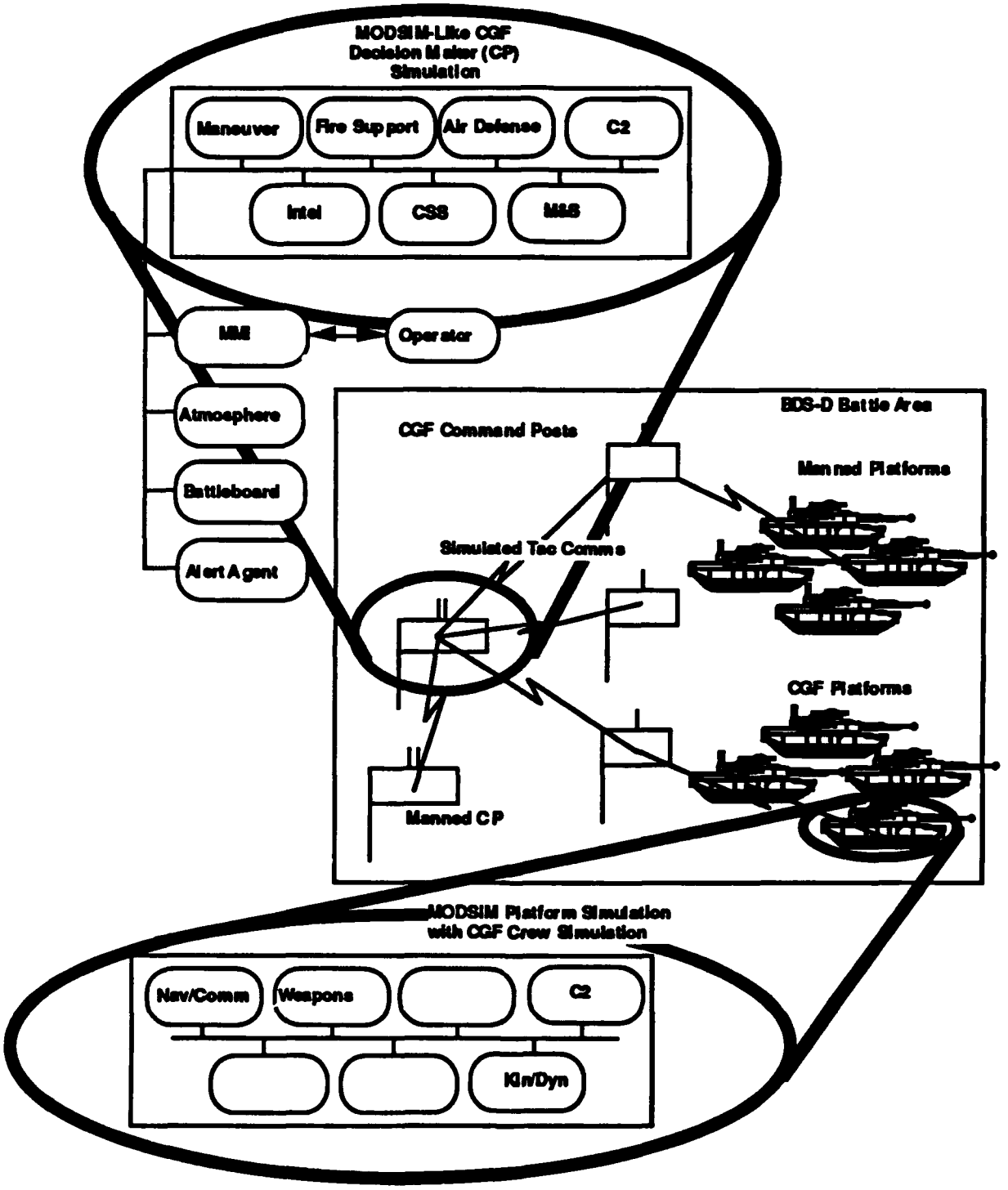


Figure 3 **The CGF Operational Architecture for DIS integrates manned and simulated command posts with manned and CGF platforms. The CGF simulates realistic unit level behavior and platform behaviors based on TRADOC Blueprint of the Battlefield functions and manned simulator performance. An explicit simulation of tactical communications supports realistic C3I.**

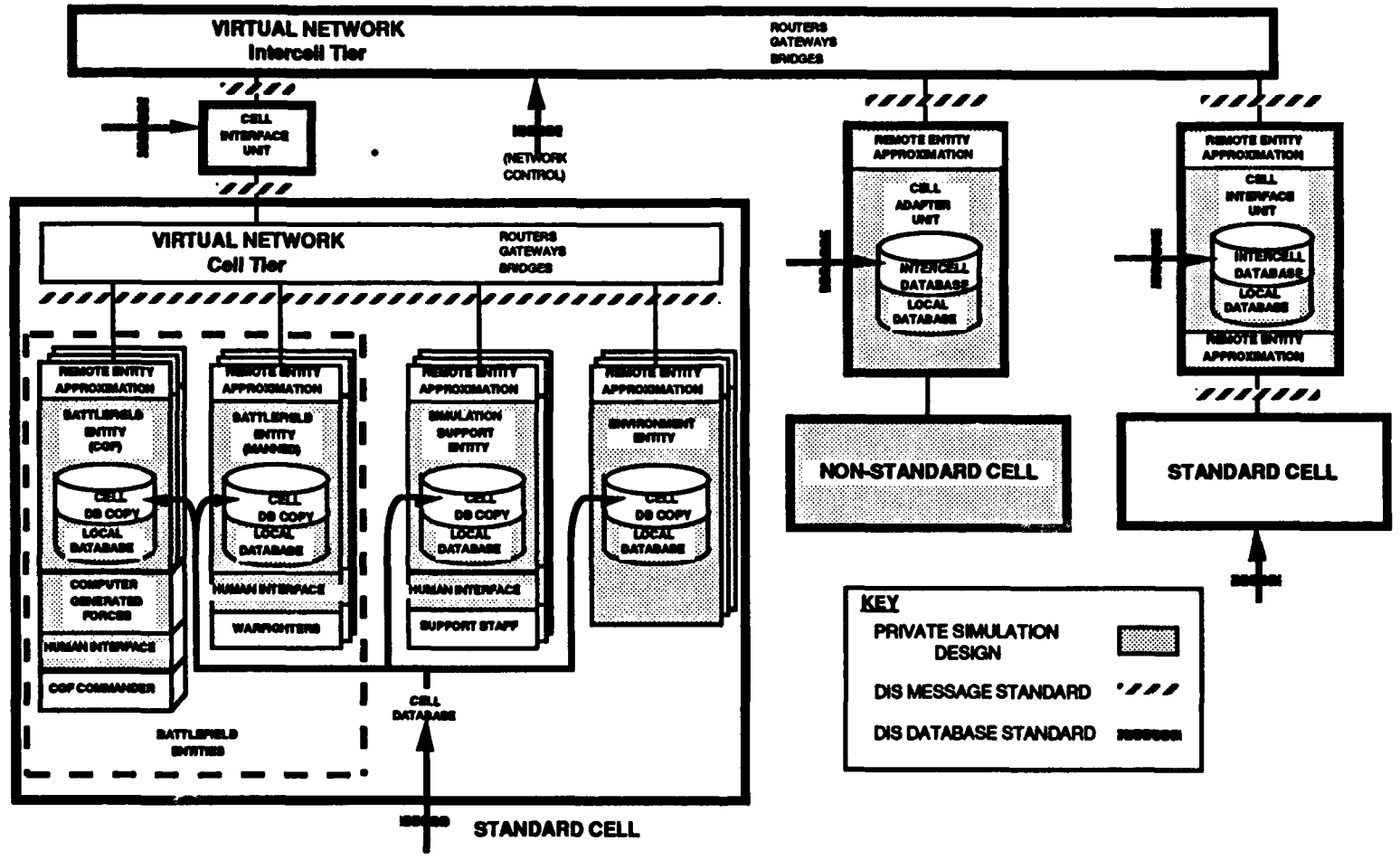


Figure 4 **Step 1 Study DIS Architecture** is a two tier network connecting heterogeneous cells, each of which contains homogeneous simulation nodes. The Computer Generated Forces (CGF) are a part of the Standard DIS Cell architecture. Multiple CGF entities within a standard DIS Cell interact with manned platforms within that cell, and with other standard cells over the cell tier of the network. In this first phase of the architecture the CGF is implemented as command post objects (and platforms), with the command posts having their internal BOS objects private as far as the DIS network is concerned.

3.2. System Architecture for CGF Demonstration

Figure 5 shows the system architecture of the CGF node of Figure 4 in a standard DIS Cell. In order to implement such an architecture, a general model of human battlefield behavior is recommended based on Action (platform) and Cognition (decision making) components. These are combined into Battlefield Operating Systems defined by TRADOC's Blueprint of the Battlefield [Pam 11-9] (and similar doctrinal functions defined by the other services). A Tactical Communications simulation component of the C2 BOS is explicitly required to provide realistic cooperative behavior between CGF units and realistic C2 of the CGF units by human commanders (see Figure 3). All CGF decision nodes interact via this tactical communications simulation. An alert agent simulation is required to provide warning to the operator when the code is insufficient to deal realistically with a situation, and to control the level of loading on the CGF operator.

Extant government funded SIMNET code modules should be used as platform and small unit drivers as appropriate in order to build on previous investment and to transition this investment into the open domain. A connection object is used to handle message passing within the CGF processes, and via the relay object to the DIS network (cell tier, see Figure 4) using a DIS network driver. Thus use of SIMNET code modules and connectivity to the DIS (cell tier, Figure 4) are decoupled into independent technical decisions. All objects should be heavily data driven, and be distinguished from each other by data files.

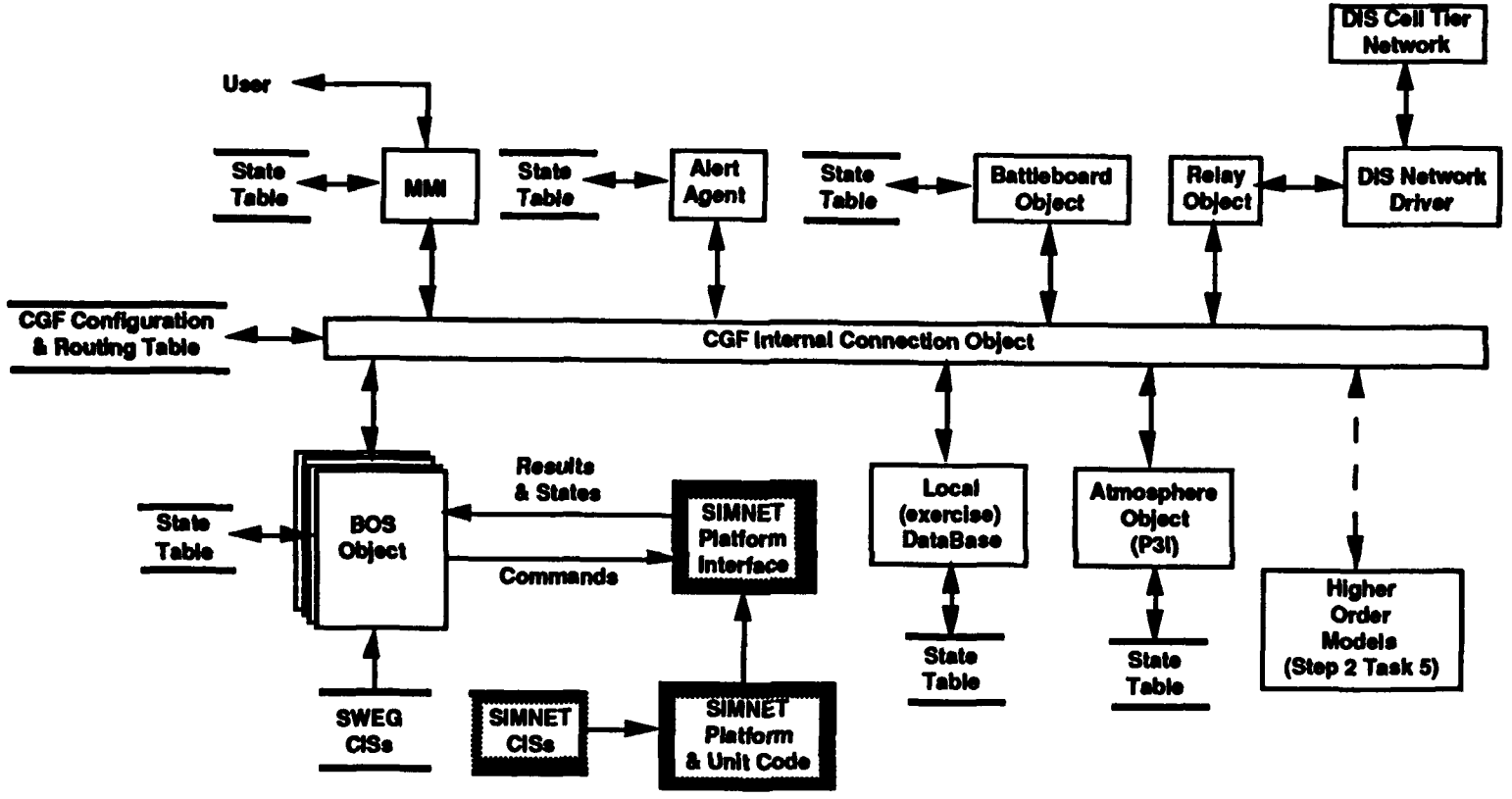


Figure 5 **The CGF System Architecture** fits within the Step 1 DIS Architecture (see Figure 3). It is based on an object oriented decomposition of the battlefield entities into platforms and decision nodes. Decision nodes (the C3I structure of the battlefield) are composed of Battlefield Operating Systems (BOS) objects [TRADOC Pam 11-9]. SIMNET SAFOR code modules are embedded in the BOS modules as appropriate. Additional behaviors and behavior combinationare provided by the Simulated Warfare Environment Generator (SWEG) which decomposes BOD functions into a general model of human behavior on the battlefield based on Action and Cognition modules. An alert agent object, inherited by every CGF decision maker, warns the CGF user when the CGF decision making code requires human intervention. Connectivity to the DIS network (the Cell Tier) is provided via a DIS interface driver.

3.3. Distributed CGF

The Step 1 Study recommends that the CGF architecture is implemented in two major phases in order to control the impact of the CGF on the DIS message protocol standards

In Phase 1 the CGF decision nodes are implemented in an object oriented fashion (based on the BOS objects), with each decision node an entity whose internals (the BOSs) are private as far as the DIS network is concerned. The CGF decision nodes interact with each other, with CGF platforms, and with manned command posts and platforms via the simulation of tactical communications. Note that the existence of a CGF tactical communication simulation has an effect on the DIS message protocol standard. CGF tactical communications must contain data representations of the message for receipt by CGF software decision makers as well as digitized speech (for receipt by human platform crews). The internals of the CGF decision making node (e.g. command post) should be implemented in an object-oriented fashion to facilitate code reuse, sharing and reconfiguration.

In Phase 2 The CGF BOS objects are distributed over the DIS networks, thus permitting geographically distributed BOS developers and users to combine their objects into CGF decision nodes (see Figure 6). If the BOS objects are distributed over the network (either intra-cell tier or inter-cell tier) then the DIS message protocol will need an entire new class of standards to handle intra-CGF node (inter-BOS) messages. Note that inter-BOS communications is often a simulation of the face-to-face communications by the people being simulated by the CGF.

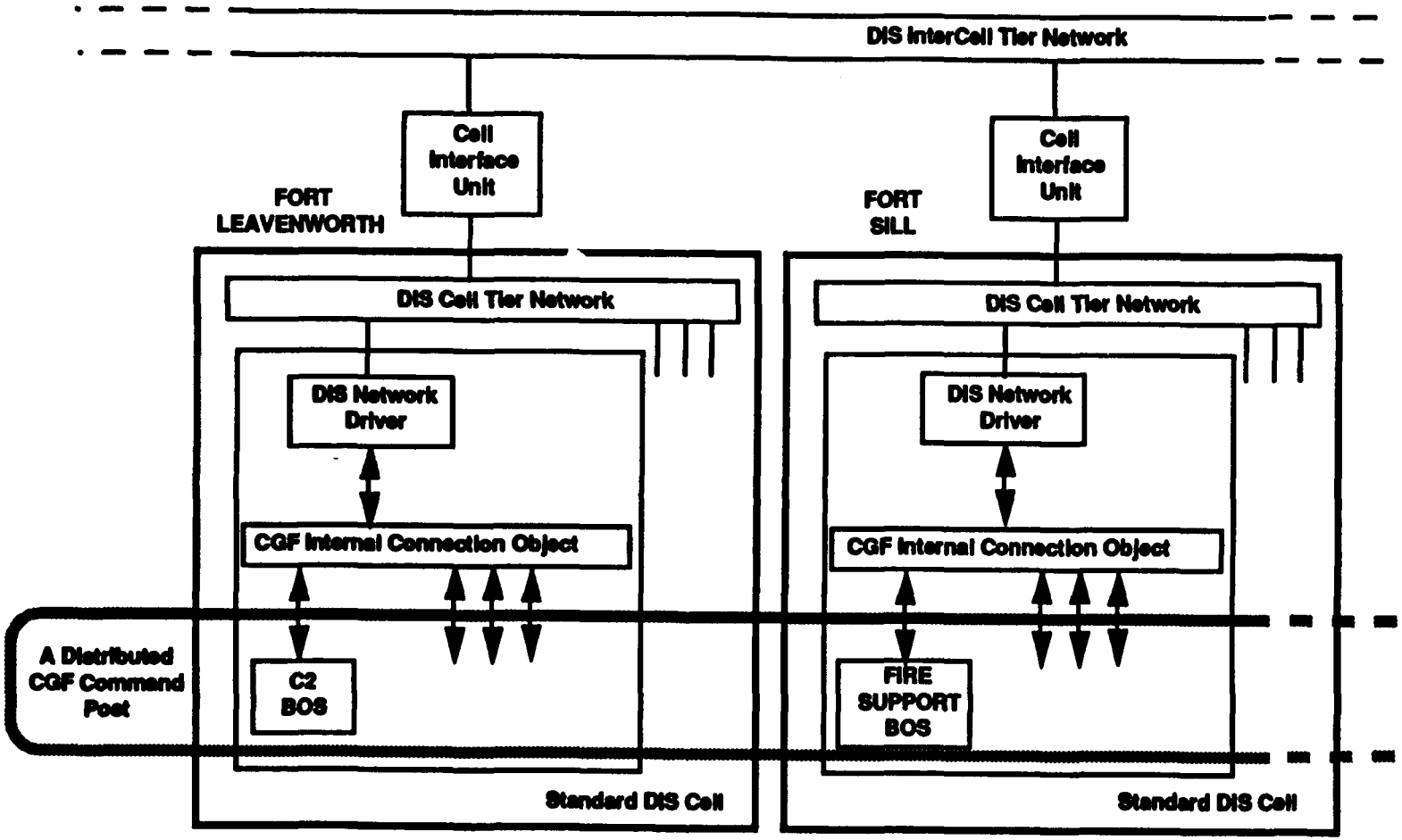


Figure 6 A Distributed CGF Architecture

A Distributed CGF Architecture creates CGF Command Posts by combining Battlefield Operating Systems provided by different organizations. In this example, the C2 BOS is provided and run from Fort Leavenworth, Fire Support by Fort Sill, and the other BOS similarly by other organizations. The implications on the DIS Message Protocol Standard are profound. Inter-BOS communications within a Command Post are a simulation of the face-to-face communications that normally occur between CP staff, and is a point to point broadcast since such communications are localized within the CP. This simulated communications must be supported by the DIS message protocol.

4 Open CGF Technologies

4.1. Battlefield Operating System Models

The DIS system is expected to deal with all the battlefield functional areas in an integrated fashion. The main technical implications of this operational objective is that the CGF software must now exhibit organized behavior in addition to that imposed by the human commander. This raises the numbers of types of behavior dramatically beyond that dealt with by current CGF technologies. A technical innovation must be implemented that supports a potentially infinite number of behaviors without requiring massive programmer resources. These behaviors must be based on doctrinal analyses of the battlefield, and be flexible enough to support changes to such analyses as the world changes. One example of a critical behavior is the ability to represent C3I behaviors, since this type of behavior is essential to battlefield functional integration.

The CGF should contain explicit simulations of decision making objects (such as command posts) organized according to US and Threat Task Organizations. These decision making objects should be developed in a modular fashion, with component objects being the Battlefield Operating Systems (BOS) defined by TRADOC's Blueprint of the Battlefield (and other doctrinal representations defined by the other services). The Decision Making objects communicate with each other, with CGF platforms, and with CGF operators via a simulation of the tactical communications system contained in the Command and Control (C2) BOS (see Figure 3). Thus CGF Decision Making Objects (such as CPs) are simulators implemented using a modular approach analogous to MODSIM. Each CGF decision making object is a combination of the doctrinal BOS objects, where each BOS object is modified by data files which depend on the type, size, and color (US, Russian or other threat) of the unit.

The recommended CGF Architecture approach tracks the seven Battlefield Operating Systems (BOS) from the Blueprint through their subfunctions to all 365 designated tasks. In parallel with this, a command level hierarchy is explicitly structured which accounts for both the action and cognitive elements of the BOS and how they interact to form the specific unit and platform objects represented on the virtual battlefield. All CGF objects are built on combinations of the small number of doctrinal Battlefield Operating Systems and their functions and tasks. There are only a small number of BOS objects and tasks, and so implementation at this level becomes low risk. However, by combining these BOSs and tasks it is possible to generate the full range of battlefield functions once the data that defines each BOS and task have been provided. Each of the CGF Battlefield Operating Systems and their tasks (and subtasks) are then decomposed into the Action (physical processes) and Cognition (decision making) components described above. This approach effectively couples the tactical vocabulary of the training and doctrine world with an artificial intelligence description of the objects on the battlefield. This tight coupling allows the CGF operators to work in

their normal regime of maps and tactical units while accessing the full power of an Artificial Intelligence object oriented approach.

The specific BOS objects within a CGF Decision Making Object (e.g. CP) interact by object based message passing, analogous to manned vehicle broadcast communications. Each BOS object is developed independently, and only once. Different levels and types of CGF Decision Making Object (e.g. CP) (including distinctions between US and Threat) are implemented by data files which control each BOS. Any CGF Decision Making Object is implemented using the same structure (as shown in Figure 7) but with different levels of logic and with different data specifications for each BOS object. Code is shared between BOS components, reducing development time, costs, and risks.

The CGF Decision Making Object has a Transport Relay Object for connectivity to the DIS network, providing interaction with the rest of DIS. In addition local copies of Battleboard and Atmosphere data base objects are maintained, the Battleboard object being a topographical representation of terrain suitable for automated reasoning and map like display. It contains networks of linear features (roads, rivers, ridge lines, valleys for example), area features (hills, lakes for example), obstacle breaches (bridges, passes, choke points for example) instead of the polygon lists contained in the vehicle terrain data base. The Battleboard and Atmosphere objects are used by the C2 BOS to decide on communications reception, and by Maneuver to plan movement, for example. An Alert object, available to any decision maker, is responsible for alerting the human decision maker when a situation occurs in which the code is incapable of realistic human behavior. This object is extremely important and so is discussed in detail later.

This approach permits interchangeability between human and CGF software decision makers and the generation of large numbers of realistic unit behaviors based on combinations of a finite number of BOS functions (this is due to the many ways functions and tasks from different BOS can be combined).

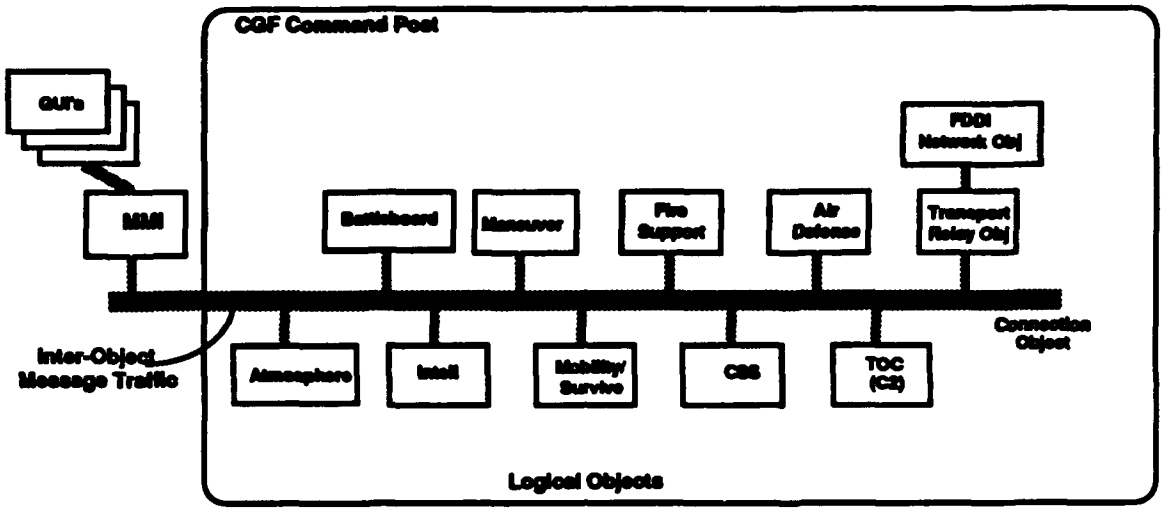


Figure 7 **A CGF Unit is MODSIM-like. It is simulated as a modular command post object with TRADOC defined BOS components, linked via the C^2 BOS with its subordinate and supporting units and platforms.**

4.2. Tactical Decision Making

CGF Capabilities are the behaviors and reactions available to the CGF units and platforms. These are generated from combinations of behaviors generated by individual BOS objects that make up the CGF Decision Making Objects, and executed by the human crew simulations using vehicle characteristics generated by the implemented manned simulators specifically for CGF vehicles. The CGF capabilities are generically defined by the capabilities programmed into the BOS objects, the complete set being all feasible combinations of behaviors emanating from each BOS for each unit.

The CGF Unit and Platform capabilities and functions are explicitly modeled from the TRADOC Blueprint of the Battlefield. This generates seven Battlefield Operating Systems, with subordinate level algorithms (see Figure 8 for a partial enumeration of the behaviors):

- Intermediate level coordinating algorithms.
- Lowest level behavioral algorithms.

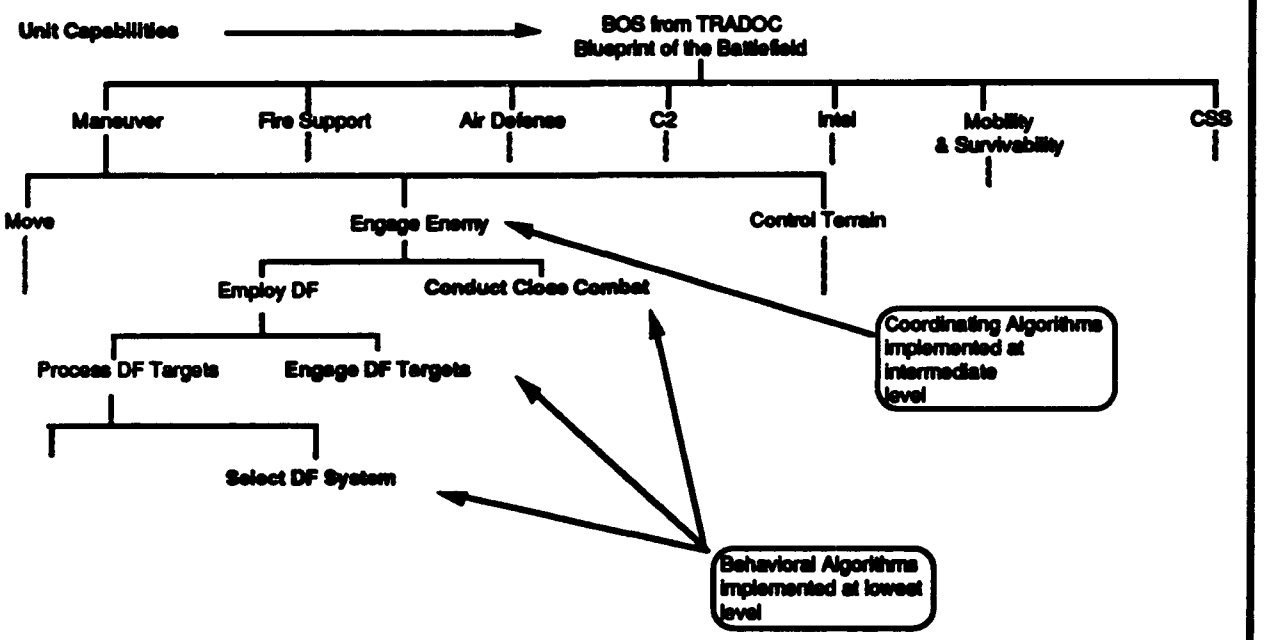


Figure 8 **CGF Behaviors are constructed from TRADOC's Blueprint of the Battlefield by combining subfunctions and tasks derived from each of the seven Battlefield Operating Systems. This generates a large set of flexible behaviors from a constrained set of well enumerated functions. This diagram shows the seven BOS and part of the Maneuver BOS down to the behavioral algorithm layer. All unit algorithms, both coordinating and behavioral, will be implemented as CISs.**

The decomposition into BOS and subordinate layers of behaviors generates a militarily correct taxonomy which maps into the behaviors of units at each level represented in DIS. This provides the military user with a familiar structure to evaluate the CGF behaviors. Combining CIS representing these behavioral and coordinating algorithms ensures doctrinally correct decisions for a wide range of unit types and battlefield situations.

Furthermore, by basing the CIS on the BOS taxonomy, a common representation of higher level functions can be used for both BLUEFOR and OPFOR. Combining the functional CIS with different military priorities, different national standards, and implementing them in different combinations under different conditions allows explicit representation of different doctrines with a minimum duplication of software. A small number of algorithms can thus generate exceptional behavioral flexibility.

About twenty or thirty CIS algorithms per BOS must be implemented for each CGF type of unit (each of which will use different combinations of them). Since there are seven different Battlefield Operating Systems to represent and several of them represent similar functions, e.g. engage targets, there are at most 200 algorithms to implement. Combinations of behavioral algorithms from each BOS with unit and environmental data tables will modify the outcome and define CGF

behavior over a wide range of conditions. Thus an enormous range of possible behaviors per CGF unit (representing the reality of the battlefield) can be obtained from a constrained set of BOS behaviors.

4.3. Realistic Behaviors for the CGF

All CGF objects are built on combinations of doctrinal Battlefield Operating Systems (BOSs) and their functions and tasks. Each of these are further decomposed into Action (physical processes) and Cognition (decision making) components. The Action and Cognition components are implemented with a small number of general functions whose combinations generate the BOS tasks for the units and the crew simulation behaviors for the platforms. There are only a small number of Action and Cognition tasks (twelve in all), and so implementation at this level becomes low risk. However, by combining these tasks it is possible to generate the full range of human behaviors. CGF platform physics is mostly Action based, CGF crew simulation is a mix of Action and Cognition, and CGF command posts are mostly Cognition based.

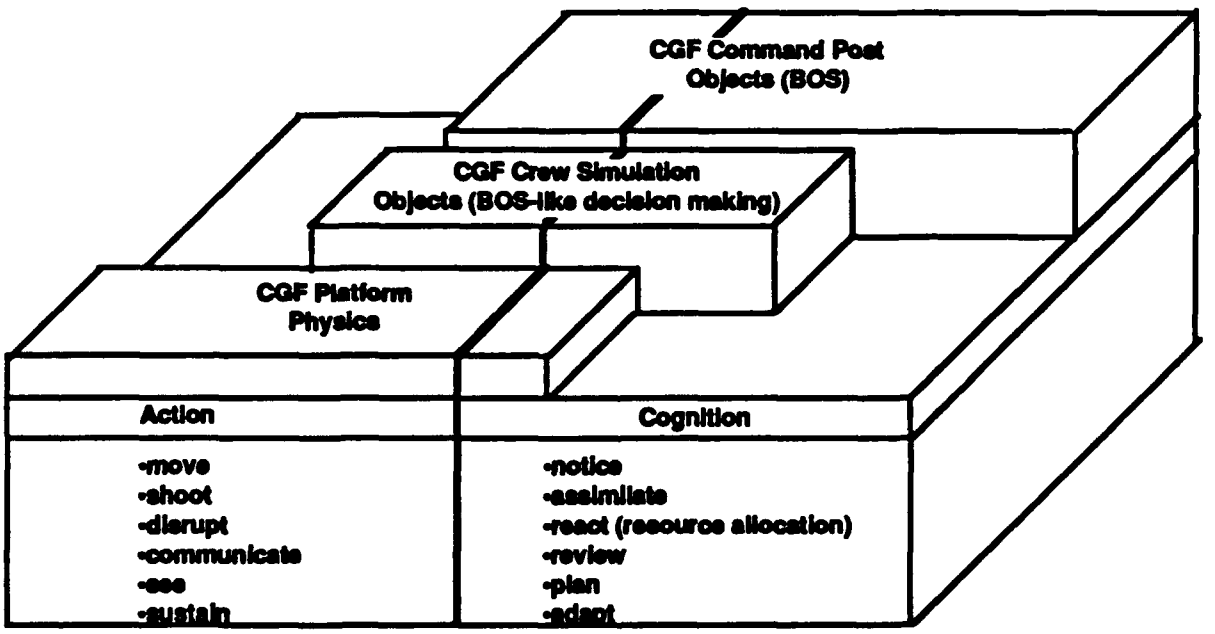


Figure 9 **Behavioral Engineering of the CGF in terms of Action and Cognition forms the foundation of the approach to realistic and flexible CGF behavior. All objects in the CGF architecture are decomposed into general Action and Cognition components, with their general purpose tasks. These general tasks are combined to provide the tasks required by TRADOC's definition of BOSs in the Blueprint of the Battlefield.**

An example of software that successfully uses this form of human behavioral engineering is the Simulated Warfare Environment Generator (SWEG) software used by the U.S. Navy at Patuxent River Naval Air Station (NAS) on the Aircraft

Combat Environment Test and Evaluation Facility (ACETEF) project and by the U.S. Navy at China Lake on research and development efforts. SWEG is a successful implementation of the Action/Cognition approach, and demonstrates the feasibility of generating large numbers of realistic and flexible behaviors from combinations of small numbers of data driven components. The Simulated Warfare Environment Generator (SWEG) can be used across a wide range of applications. It is an example of mature and robust simulation technology, and its use in modules of the CGF can be used to prove the open nature of the CGF architecture.

SWEG controls the real-time operation of assets during a test or exercise. An asset represents something, part of something, or several things, in the exercise. An asset is some combination of people, software, and hardware. A flight simulator could be an asset. A simulation of a JTIDS communication network could be an asset. A simulation of a Combat Information Center's (CIC) data fusion techniques could be an asset. SWEG can be configured to interface with these and other assets. The simulation portion of SWEG can also be an asset. SWEG can exist on multiple computers, with each copy representing a different portion of the scenario. This allows the number of entities in the scenario to expand out to the limits of the communication links and host computers.

SWEG can represent anything in the scenario that is not represented by another asset. This capability is useful during integration tests leading up to an exercise. If an asset is not available during a block of time, then SWEG can be configured to represent that asset's functions, as well as the functions that SWEG would have normally. It follows, then, that SWEG can simulate the entire scenario, at a reasonable level of fidelity, in a standalone mode. In this case, SWEG can run in real-time, at a constant rate that is faster or slower than real-time, or in a "go as fast as you can" mode. When integrated with other assets SWEG can run faster or slower than real time, depending on the capabilities of the other assets.

SWEG produces, and collects data from other assets, that can be sent to SWEG's graphics routines or other graphics packages for display in real time, near real-time, or in a post-exercise replay mode.

SWEG is independent of the computer host or communications topology. ACETEF, for example, is based on shared memory for local communications between processes, and Ethernet for communications between separated assets. It used dedicated commercial telephone lines with encryption devices to establish a real-time link with the USAF REDCAP facility in Buffalo, NY. Functionally, SWEG sends and receives information periodically or aperiodically. The update rate for periodic messages can be varied by type of message or type of asset. This eliminates the need for sending everything at the most frequently required update rate. Only the interface code must be changed if a new interface technique is employed. SWEG runs on Silicon Graphics and various DEC computers at ACETEF.

SWEG uses DMA data for terrain. It represents signatures and antenna patterns in three dimensions, with user-controlled variable amounts of granularity. It represents movement with five degrees of freedom (velocity is oriented along the longitudinal axis of the vehicle). It is expected that this limitation will disappear within the next year. Tactics and doctrine, command chains, and related aspects are contained in the data. SWEG makes no C3 assumptions. Likewise, weapon systems, jammers, sensors, movement capability, and communication systems are defined in the data, not in the code. SWEG will perform flat earth, curved earth, and effective earth radius calculations depending on the the user's data, by type of system, for all energy-related calculations.

4.4. CGF Platform Objects

All CGF platform types will be based on a modular platform simulation containing two subcomponents: the platform object which handles action items such as weapons, dynamics and kinematics; and the crew object which simulates crew decision making. The platform action object code will consist of modified versions of the MODSIM algorithms used by the DIS manned vehicle simulators (for example, CGF platforms do not need detailed internal models of the engine) driven by data tables. Where a manned vehicle simulator is not available as a basis for a CGF platform, the CGF will provide the MODSIM equivalent algorithms. The data tables required to drive CGF platforms not represented by manned simulators will be provided by empirical data drawn from combined arms and joint tests. It is these data tables that provide the physical fidelity.

All input/output to the human crew is replaced with a CGF crew simulation. The CGF crew simulation is based on the same structure as the CGF decision making object, being divided into battlefield operating systems. The BOS for platform crew will be much simpler than those for unit CPs and will be generated from a simpler set of the same BOS cognition structures. The crew simulation will command and control the platform using CISs passed through the tactical communications simulation. The MODSIM equivalent algorithms will execute the move, shoot, communicate and see actions. Data files distinguish each platform by type, capability, and color (US, Russian, or other threat).

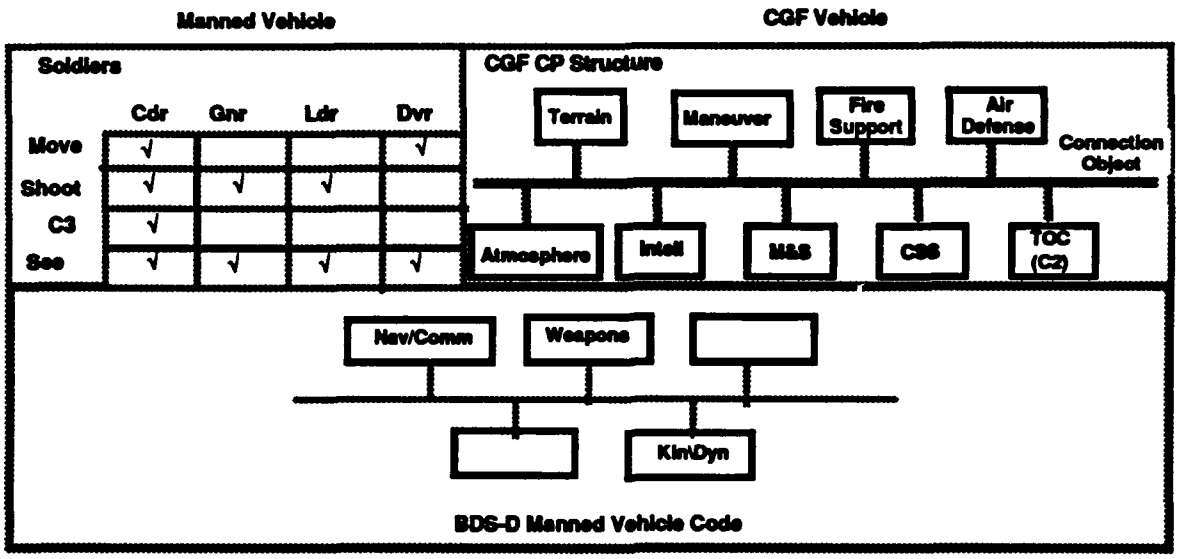


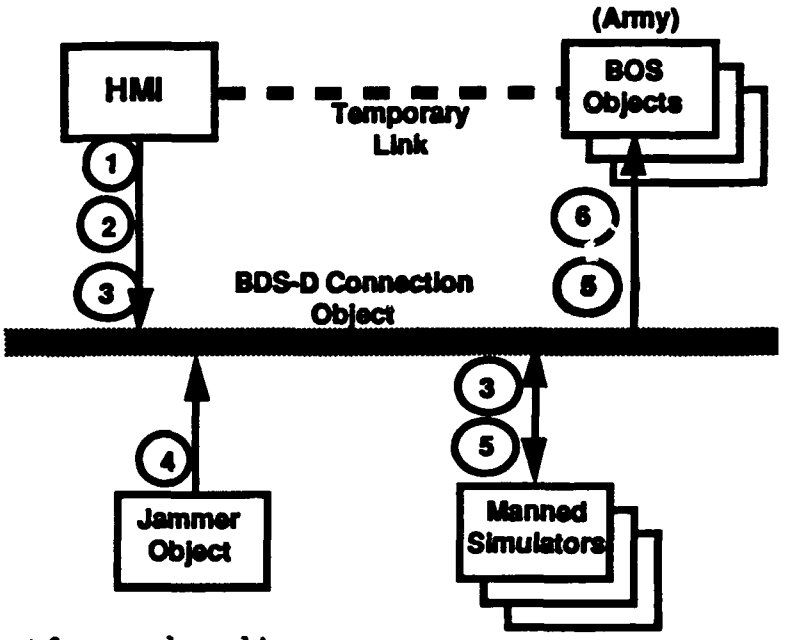
Figure 10 The CGF Platform Simulation object uses kinematics, dynamics, vulnerability, and damage data from the manned simulators running on modified manned simulator algorithms. When a manned simulator version of a required CGF platform is not available, then empirical data from joint tests is used. A CGF human crew simulation replaces I/O to the human crew, and is based on the modular BOS approach to the CGF CP.

4.5. Tactical Communications Simulation

An extremely important part of the CGF is the simulation of tactical communications, which is at the heart of intelligent cooperative battlefield behavior. In order for the CGF to behave realistically at the unit level it is necessary to simulate the flow of information and orders through the command hierarchy. This has the added advantage of allowing jamming, terrain masking and atmospheric effects to interfere with the flow of communications, thus generating realistic confusion and delays (i.e. the fog of war). Seamless Simulation issues such as electronic warfare using operational equipment integrated with DIS become possible only if the CGF explicitly simulates the tactical communications contained within the C2 BOS. Thus the CGF software decision makers will communicate with each other, with the human operators, and with their subordinate platforms via an explicit simulation of tactical communications contained within the Command and Control (C2) Battlefield Operating System (BOS) (see Figures 3, 7 and 11).

CGF generated tactical communications must be receivable by other CGF units (operated by humans or by software). The CGF Tactical Communications architecture supports direct human to CGF software tactical communications through the use of text generation from data structures, speech generation from text, and speech recognition (see Figure 11). The DIS Message Standards must

include data representations within the radio message packet in order to support CGF communications.



1: Input forms and graphics
2: Radio message generation - text generation from data structures
3: Radio message generation - digitized speech generation from text
4: Radio jamming - generate data structure and digitized noise
5: Receive / No receive decision
6: Speech recognition and understanding

Figure 11 The CGF Tactical Communications Simulation handles communications from CGF to CGF and to manned simulators. Speech generation from data by the CGF is recommended rather than generation by the receiver nodes since this permits older platform simulators to receive CGF transmissions without requiring upgrades with speech generation technology. This however is a trade-off between increased data rates on the networks created by CGF speech generation versus technology upgrades of all platform simulators.

4.6. Alert Agent

However, for reasonable expenditures of resources there will always be cases where CGF software is faced with situations it is unable to handle in a tactically realistic fashion. The traditional approach to this problem in the CGF has been to rely on the operator to monitor all CGF units and platforms, and to immediately intervene when necessary. One result of this is that the human commander is often overloaded by the CGF. This problem will get worse as the CGF expands into higher echelons, and more layers of CGF software exist between the human commander and the platform level of interaction. The behavioral engineering

approach recommended by the step 1 study will reduce the frequency of such required interventions, but will not remove them completely. Indeed, it is doubtful that complete removal is advisable, as the CGF is considerably enhanced by the insertion of human ingenuity via the intervention mechanism.

The CGF Alert Agent avoids operator overload in the face of unavoidable software requests for human intervention. In a complex and time sensitive tactical situation, e.g. where a CGF unit is being overrun, the CGF unit will respond tactically using the communications simulation. At this time the operator has the option of intervening as he would on the real battlefield, or letting the situation run its course. The CGF alerts system requests human control when parametrically defined triggers fire (e.g., casualties high, rates of advance low). The operator can opt for the default alert setting or rank order them in levels of priority. The operator also sets the alert interface to control volume of incoming alerts. On the other hand, if the situation becomes too complex for the code, then the CGF code recognizes that it has no valid default CIS and immediately alerts the human operator for help in order to avoid compromising the exercise. In this case the operator has the option of an immediate intervention by bypassing the Tactical Communications simulation. This new CGF capability (code that can identify when it cannot handle a situation) is handled by an object called the Alert Agent. It is a simple, extendible, and powerful technique available to all BOS objects.

Alerts to the operator are thus minimized, and under user control. The user can tailor the alert system to focus on specific units, events or interactions according to his preferences. Furthermore, the alert agent is used to automate the identification and logging of situations in which the CGF code cannot behave with realistically, thus providing valuable support for system upgrades.

All CGF units operate autonomously as the default. Each CIS being executed by a CGF unit carries parameters (for example a doctrinal advance rate for an attack). When the alert filter detects actual situation parameters are out of range of desired parameters (for example actual rate of advance is below doctrinal or ordered rate) an alert is generated, rank ordered, and passed to the operator or dropped depending on the threshold requested by the operator. All CISs will have alert filters associated with them, not just the examples indicated in this diagram.

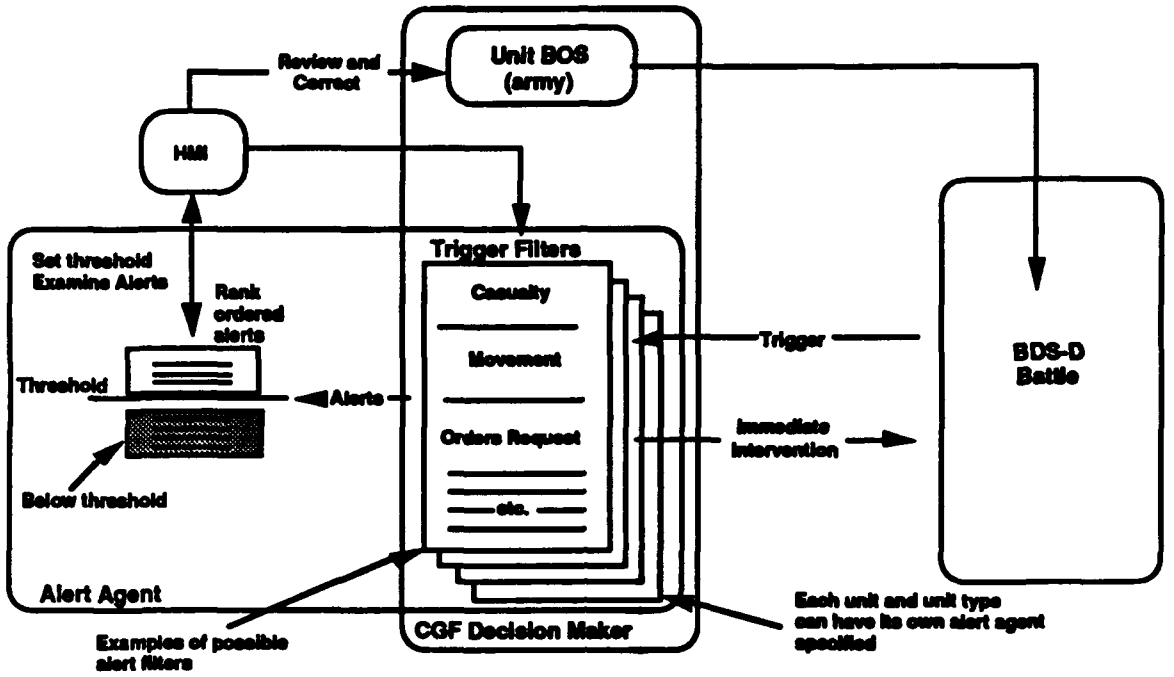


Figure 12 The CGF Alert Agent integrates the autonomous and human control of CGF units, providing a shared initiative system with significant enhancement to CGF realism and reduction in operator loading.

4.7. Senior Commander Support Tools

The C2 BOS also contains planning tasks such as Course of Action (COA) Evaluation. A CGF simulation system thus has to emulate the COA evaluation, i.e. provide an emulation of the branch wargaming carried out by the G2 and G3. The COA evaluation is a faster-than-real-time approximate simulation of the perceived situation, and should be provided as part of the CGF architecture. Since the CGF human commander will have access via the user interface to all BOS simulations, this will provide the commander with a faster than real time planning and evaluation tool for pre-battle as well as battle-time planning.

The structure of the CGF decision making object (as exemplified in the CGF Command Post Simulator) directly supports faster than real time wargaming at the unit level to provide planning support to the human trainees prior to an engagement. Every CGF Command Post has an embedded C2 BOS, which itself contains the subfunctions "develop courses of action", "analyze courses of action", "compare courses of action", and "select or modify course of action". These subfunctions collectively are the wargaming process by which the commander (or his staff) evaluate courses of action and make their recommendations. The CGF simulator will emulate this process in the C2 BOS, and this BOS is available for control by the CGF commander. The wargaming process in the C2 BOS is a faster than real time simulation of the DIS battlefield at the unit level of aggregation, the

level of aggregation being one and two levels below the command level of the command post. The CGF commander is permitted by the CGF system to take over direct control of any subordinate unit, and hence can wargame at any level of detail for all subordinate units down to the vehicle level.

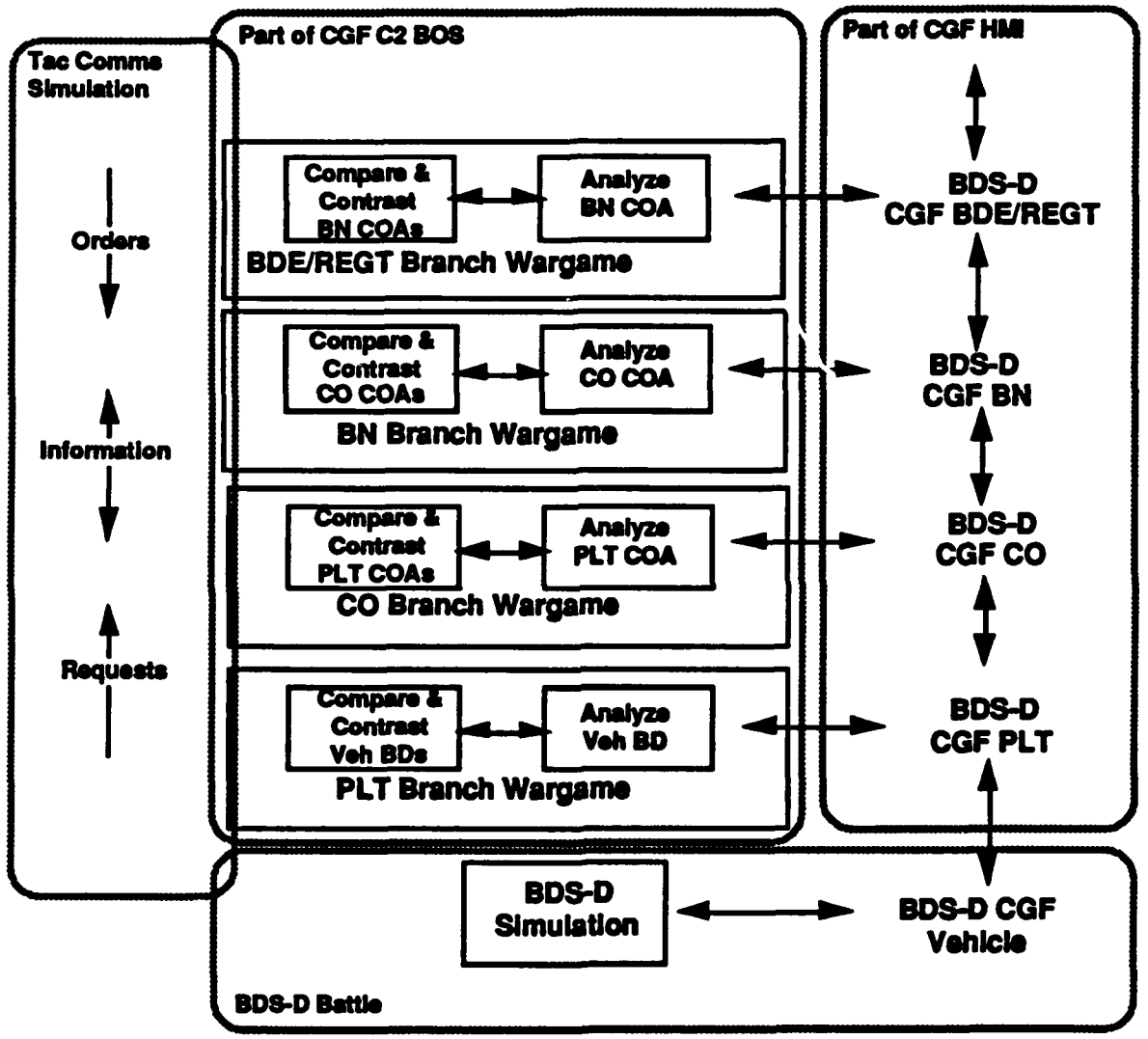


Figure 13 **Senior Commander Planning Tools are available from the BOS based CGF Command Post. For example, the C2 BOS contains an emulation of the course of action evaluation process which is available to the CGF commander via the MMI.**

4.8. User Interfaces

The CGF must support two broad uses. First it must provide a component of the DIS environment for human CGF commanders as warfighters "fighting to win". In this use, the CGF is a tactically realistic user interface into the DIS for

military commanders. Second it must provide control of the DIS environment for trainers and analysts by generating and manipulating the CGF. In this use the CGF becomes a tool for direct manipulation of the DIS environment. Both these uses must be supported by the same user interfaces and simulation system, without requiring complex restructuring of the CGF system when switching between uses. Furthermore, the user interface must permit the operator to take command of any software decision making node within the simulated C3I hierarchy of the CGF. Plug-in compatibility between operator and software is required to provide a flexible insertion of human ingenuity without overloading the operator or generating areas of the CGF which are outside human control.

The difference between the tasks of the Warfighter "fighting to win" and the Analyst/Trainer "controlling the environment" are characterized by the degree of penetration into ground truth and the degree of control over the environment, as shown in Figure 14. The Warfighter perceives the DIS battlefield and controls his forces through a Tactical Communications simulation system with realistic unreliability and by moving around in a manned simulator vulnerable to combat damage. The Trainer/Analyst, however, has perfect Tactical Communications, can exercise immediate intervention in controlling CGF forces, and can travel the DIS battlefield in the on-screen Stealth Vehicle. These very different working environments can be distinguished entirely by providing each with different privileges and levels of control over the Tactical Communications simulation and the Alert Agent components of the user interface. The warfighter has to use the complete tactical communications simulation, with realistic delays, jamming, lost messages etc., in order to communicate with and control his forces, and he has restricted access to the alert agent. The trainer/analyst however is granted perfect tactical communications with immediate and perfect message delivery, and has access to a complete alert agent with full immediate intervention privileges

In addition, the Trainer/Analyst is provided a Stealth and Plan View Display while the Warfighter uses manned platform simulators and simulated intelligence displays. This increases the flexibility of the overall DIS CGF system since with the proper password and initialization data the same CGF system can serve dual purposes. The use of a common MMI with identical maps, screens, and windows (with some functions simply restricted to one type of operator) ensures that a consistent MANPRINT standard is used. The common MMI also reduces the effort to develop, maintain, and enhance separate screens.

	User Resources			Impact on	
User	Tac Comms	Alert Agent	Platform	C2	Intelligence
Warfighter	Subject to battlefield degradation	Limited privileges.	Manned simulator subject to physical constraints and vulnerable to combat damage.	Constrained by Tac Comms simulation, subject to battlefield degradation.	What is seen, reported and deduced.
Trainer/ Analyst	Perfect	Complete privileges.	Stealth Vehicle with no physical constraints and no vulnerability.	Immediate, on demand	Ground truth

Figure 14 The CGF Architecture Supports Dual-Use by using different access privileges to the same user interface components. This ensures that duplication of effort does not occur when supporting different user types.

The MMI screens should be developed using a commercially available Graphical User Interface (GUI) builder, and should use COTS producst such as X windows and Motif for portability. An open approach to interfacing the MMI screens with the underlying CGF functionality will permit development of the MMIs separate from the rest of the CGF, and will facilitate the future development of specialized MMIs for particular end-users who may wish a different MMI for their purposes (although we expect that most users will be satisfied with the ability to alter the MMI via privilege control as discussed).

MMIs should be developed in conjunction with Subject Matter Experts (SMEs) so that the MMI accurately reflects what will be most effective in real world use. Consultation with these SMEs will also ensure that privileges and displays are proper for the type of user (e.g. Warfighter vs. Trainer).

Multiple tactical units, BOSs and platforms (as well as Battleboard or Atmosphere objects) can be presented to a single operator for monitor and control (depending on operator loading constraints), but each such object can have only a single human controller (see Figure 15). Each CGF unit attached to a specific operators Man-Machine Interface (MMI) has logical links to one or more BOS objects. At an operator request, each unit or BOS can display its state and the state of its subordinates either as seen from its CP object (perceived truth) or from the platform objects (ground truth).

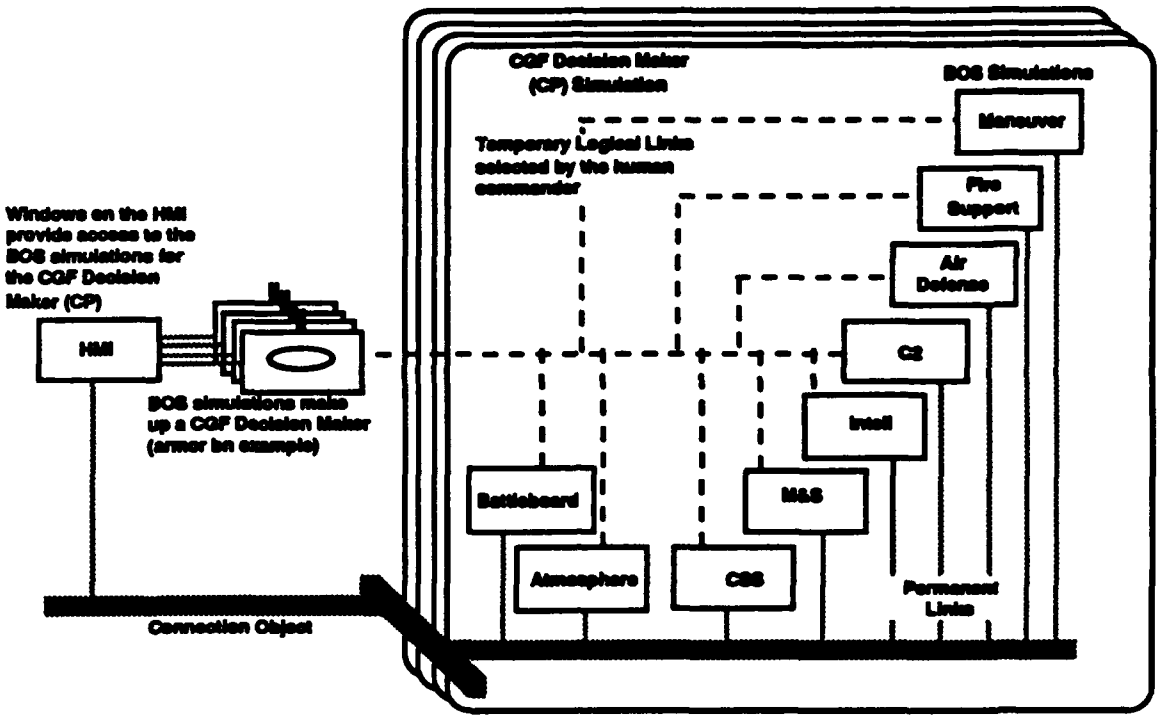


Figure 15 CGF Operator Interface allows the operator to interface to any BOS in a CGF Unit for direct control or supervision.

Human CGF operators will input orders and messages to CGF units at any echelon via the MMI using map graphics, menu selections and and forms. The MMI will translate these into the internal machine representation of the CISs. Alternatively, CGF software will generate the CIS internal representation directly. Both processes generate radio message which are passed to the tactical communications simulation within the C2 BOS for simulated transmission over the BDS-D network in the form of DIS PDUs containing the CIS structures. At the receiver, a receive/no receive decision will be made based on transmission and receiver characteristics and intervening terrain and atmospheric conditions. If a receive decision is made, then the receiving CGF software operates directly on the CISs, and any MMI at that location translates the CISs into the map graphic and text/forms representation for the human operator (see Figure 11). The CGF operator has the option to enforce the immediate arrival of a message to provide direct control of a vehicle or to ensure that a command is acted upon by a higher echelon unit.

5 Potential Additional Areas of Research

The CGF system must be capable of easy upgrade and expansion, and it must be implemented such that new requirements and technologies can be inserted. The structure of the recommended CGF architecture easily permits such extensions because of its modular structure and the data dominant nature of the software which allows many changes to be implemented in data with minimal software changes. Figure 16 provides an overview of potential future requirements, the associated technology extensions, and the specific action which will be taken with CGF to support implementation of the technology extensions.

Future DIS Requirement	Technology Extensions	CGF support for the Technology Extensions
Implement in Ada and produce production quality documentation on a module by module basis.	Object Oriented Extensions to Ada.	Modular architecture supporting independent modification to selective components.
Integrate with operational equipment. Integrate with other simulation nets such as ALSP. Integrate with Higher Order Models.	Seamless simulation.	Distributed architecture and standard network protocol.
Distributed CGF.	Extend the DIS Message protocol to include inter-BOS messages on the CGF connection object, and distribute individual CGFs over the DIS network.	Modular CGF architecture based on message passing.
Support senior command training.	Scale to higher echelons Increase numbers of vehicles Faster than real time Decision tools.	Explicit BOS based CGF architecture, distributed architecture. CGF C2 BOS supports higher level unit decision making.
Tac Comms between CGF and manned platforms.	Speech generation Speech recognition and understanding.	The CGF C2 BOS explicit Tac Comms simulation.
Autonomous CGF intel.	Speech recognition.	Explicit Tac Comms simulation.
CGF learns automatically.	Neural networks. Genetic algorithms. Synthetic annealing. Naive physics.	Each BOS can be replaced or upgraded independently.

.c.Figure 16 CGF Technology Extensions are driven by future DIS requirements; and are supported by the CGF structure. In particular, all BOS simulations are constructed independently and can thus be upgraded by different users. The Tactical Communications simulation permits speech recognition extensions, and upgrades to intelligence and electronic warfare, by integration with operational equipments.

A major technology extension for the CGF are fully autonomous C3I BOS objects, thus supporting integration of human and software decision making at all levels of command and control including at the platform level. A critical element of this is the ability of CGF software to receive and respond to radio messages from manned simulators and command elements. This is only possible with the incorporation of speech recognition and analysis algorithms. The CGF architecture supports the incorporation of these technologies into the CGF Tactical Communications Simulation as shown in Figure 11.

6 Step 1 Task 1 Study Conclusions

The Computer Generated Forces is a necessary and critical component of BDS-D. It is required to provide:

- large numbers of DIS forces with low manpower requirements.
- simulation of the performance and appearance of manned simulators.
- simulation of the performance and appearance of platforms for which no manned simulators exist.
- simulation of human decision making in a simulated C3I hierarchy.
- user interfaces to any decision making node in the simulated C3I hierarchy for the CGF.
- provide credible and realistic large scale battlefield behaviors.

It is currently treated as a single large system, implemented by a single developer. The CGF however is sufficiently large and complex that it warrants its own architecture with components capable of independent development, modification, and expansion by multiple users. Such an architecture will decouple the risks associated with each technical area of the CGF, and permit a reconfigurable CGF capable of changes without the major rewrites that have plagued the current system. Furthermore, most DIS participants will want to insert their systems into a large scale and rich simulated battlefield environment, but will not have the resources to provide that environment. Therefore the step 1 study determined that the Computer Generated Forces is an integral part of the DIS infrastructure, and is thus part of the DIS architecture.

The CGF architecture must provide within a modular and open system user interfaces for different purposes (developers/modifiers, warfighters, trainer/analysts, et cetera), realistic battlefield behaviors, and expandability in scale, scope and realism. The critical technical problems are to provide general models of human battlefield behavior which can be combined into reconfigurable models of battlefield functions without the risks normally associated with attempting to simulate human behavior. These models must generate credible and realistic behavior, and be scaleable to large battlefields without prohibitive personnel or hardware costs. The architecture must be such that the government can call upon multiple independent developers (industrial, academic, military and government teams) to develop, modify, expand and maintain the CGF components. It should make maximum use of current industry standards, and avoid as much as is possible unique solutions and technologies.

The CGF Architecture must support the operational objectives of the CGF and deal with the problems identified by the step 1 study. The step 1 study concluded that a modular CGF architecture is feasible, and recommend that a proof of

concept be built whose goal is to demonstrate an effective CGF capable of modular and open implementation and modification. Within the overall goal of providing a modular and open CGF, the following components of an architecture are viewed as critical in order to deal with the problem areas identified by the step 1 study:

- User Interfaces
- Realistic CGF Behaviors
- Alert Agent for the CGF
- Incorporation of all Battlefield Functions for all Services
- Modular transition to Ada
- Integration with Higher Order Models
- Distributed CGF
- Senior commander decision support tools
- Adaptive CGF behaviors

The Step 1 Study recommends that the CGF architecture is implemented in two major phases in order to control the impact of the CGF on the DIS message protocol standards. In Phase 1 the CGF decision nodes are implemented in an object oriented fashion (based on the BOS objects), with each decision node an entity whose internals (the BOSs) are private as far as the DIS network is concerned. The DIS message standard must be extended to handle data structures within the radio message packet, thus permitting CGF to transmit and receive simulated radio messages. In Phase 2 the CGF BOS objects are distributed over the DIS networks, thus permitting geographically distributed BOS developers and users to combine their objects into CGF decision nodes. In this phase, the DIS message standard must be extended to handle inter-BOS coommunications, which simulate face-to-face human communications.

Acknowledgements

The author wishes to thank Charles Burdick and Peter Lattimore of BDM International, Timothy Eller of LORAL WDL, and John Morrison and Warren Katz of MaK Technologies for their critical and constructive comments concerning the CGF Architecture. The author particularly thanks Peter Lattimore for the information on the Simulated Warfare Environment Generator (SWEG) described in section 4.3. Peter Lattimore is the developer of SWEG.

Appendix B: References/Bibliography

Abrett, G., Deutsch, S., Downes-Martin, S., 1990a

Guided Planning and Plan Execution. Presented at the AAAI Workshop on Automated Planning for Complex Domains, 1990.

Abrett, G., Deutsch, S., Downes-Martin, S., 1990b

Two AI Languages for Simulation. Transactions of the Society for Computer Simulation, Vol. 7, No 3., 1990.

Booch, G., 1991
Object Oriented Design with Applications. Benjamin/Cummins 1991.

Bach, W., 1989
Is Ada Really and Object-Oriented Language?. Journal of Pascal, Ada & Modula Volume 8, Number 2, March/April 1989.

Brooks, R., Buchanan, B., Lenat, D., McKeown, D., Fletcher, J., 1989

Panel Review of the Semi-Automated Forces. Institute for Defense Analyses Document IDA D-661, September 1989.

DARPA, 1991b

Battle of 73 Easting: Executive Summary. Distributed at the DARPA 73 Easting Conference 27 to 29 August 1991.

DEC, 1991
Application Control Architecture: General Information Guide. Digital Equipment Corporation, 1991.

Deutsch, S., 1989

Tactical Action Representation Language (TARL) Components. In [Brooks et al, 1989].

Downes-Martin, S., Saffi, M., 1987

SIMNET Semi-Automated OPFOR: A Functional Description (Version 1). BBN Report No. 6555, 1987.

Downes-Martin, S., 1989a

Semi-Automated Forces: Concepts and Approach. In [Brooks et al, 1989].

Downes-Martin, S., 1989b

Semi-Automated Forces: Combat Simulation for the Future. In Proceedings of the 57th Symposium of the Military Operations Research Society, 1989.

Downes-Martin, S., 1991

Seamless Simulation Literature Survey. Prepared for the Institute for Defense Analyses, November 1991

Downes-Martin, S., 1991
The Combinatorics of Vehicle Level Wargaming for Senior Commanders. In [IST, 1991c].

Garvey, R., Monday, P., 1989

SIMNET (Interactive SIMulator NETworking). Phalanx, 22, 1, March 1989.

GTRI, 1990

Modeling and Simulation in Training. Georgia Institute of Technology Education Extension Course, April 1990.

IST, 1989

State-of-the-Art Assessment for Simulated Forces. Technical Report IST-TR-89-18, Institute for Simulation and Training, University of Central Florida, November 1989.

IST, 1990a

Summary Report: The Second Conference on Standards for the Interoperability of Defense Simulations. Volume I: Minutes. Technical Report IST-CF-90-01. Institute for Simulation and Training, University of Central Florida, January 1990.

IST, 1990b

Summary Report: The Second Conference on Standards for the Interoperability of Defense Simulations. Volume II: Attendees List and Viewgraphs. Technical Report IST-CF-90-01, Institute for Simulation and Training, University of Central Florida, January 1990.

IST, 1990c

Summary Report: The Second Conference on Standards for the Interoperability of Defense Simulations. Volume III: Position Papers. Technical Report IST-CF-90-01, Institute for Simulation and Training, University of Central Florida, January 1990.

IST, 1990d

Rationale Document: Entity Information and Entity Interaction in a Distributed Interactive Simulation. Technical Report IST-PD-90-1, Institute for Simulation and Training, University of Central Florida, June 1990.

IST, 1990e

Military Standard (Draft): Entity Information and Entity Interaction in a Distributed Interactive Simulation. Technical Report IST-PD-90-2, Institute for Simulation and Training, University of Central Florida, June 1990.

IST, 1990f

Summary Report: The Third Workshop on Standards for the Interoperability of Defense Simulations. Volume I: Minutes from Plenary Session and Attendees List. Technical Report IST-CF-90-13, Institute for Simulation and Training, University of Central Florida, August 1990.

IST, 1990g

Summary Report: The Third Workshop on Standards for the Interoperability of Defense Simulations. Volume II: View-Graphs from Plenary Sessions. Technical Report IST-CF-90-13, Institute for Simulation and Training, University of Central Florida, August 1990.

IST, 1990h

Summary Report: The Third Workshop on Standards for the Interoperability of Defense Simulations. Volume III: View-Graphs from Working Groups Presentations. Technical Report IST-CF-90-13, Institute for Simulation and Training, University of Central Florida, August 1990.

IST, 1990i

A Testbed for Automated Entity Generation in Distributed Interactive Simulation. Technical Report IST-TR-90-15, Institute for Simulation and Training, University of Central Florida, August 1990.

IST, 1990j

Proceedings of the First Behavioral Representation and Computer Generated Forces Symposium: A DARPA Research Initiative. Institute for Simulation and Training, University of Central Florida, October 1990. Goldiez, B. (Ed).

IST, 1991a
Military Standard (Draft): Entity Information and Entity Interaction in a Distributed Interactive Simulation. IST-PD-90-2 (Revised), Institute for Simulation and Training, University of Central Florida, January 1991.

IST, 1991b

Rationale Document: Entity Information and Entity Interaction in a Distributed Interactive Simulation. IST-PD-90-1 (Revised), Institute for Simulation and Training, University of Central Florida, February 1991.

IST, 1991c
Proceedings of the Second Behavioral Representation and Computer Generated Forces Symposium: A DARPA Research Initiative. Institute for Simulation and Training, University of Central Florida, May 1991. Mullally, D., Petty, M., Smith, S. (Eds).

IST, 1991d

Summary Report: The Fourth Workshop on Standards for the Interoperability of Defense Simulations. Volume I: Minutes and Attendees List. Technical Report IST-CR-91-11, Institute for Simulation and Training, University of Central Florida, March 1991. McDonald, B., Bouwens, C., Carr, D. (Eds).

IST, 1991e

Summary Report: The Fourth Workshop on Standards for the Interoperability of Defense Simulations. Volume II: Position. Technical Report IST-CR-91-11, Institute for Simulation and Training, University of Central Florida, March 1991.

Katz, A., 1992

Absolute Time Stamp in Networking of Simulators. Position Paper to the DIS Standards Committee, Institute for Simulation and Training, Unversity of Central Florida, Orlando, FL., March 1992.

McBride, D., 1990

DARPA Research Objectives. In [IST, 1990j].

OMG, 1991a
First Class, The Object Management Group Newsletter, September/October 1991.

OMG, 1991b

The Common Object Request Broker: Architecture and Specification. Object Management Group Document 91.8.1, August 1991. Digital Equipment Corporation, Hewlett-Packard Company, HyperDesk Corporation, NCR Corporation, Object Design Inc, SunSoft Inc.

Pascoe, G., 1986
Elements of Object-oriented programming. Byte, August 1986.

Pasha, 1991a

War Training and C4I ops may be joined. C4I Report, Vol. 6, No. 3, February 4, 1991.

Pasha, 1991b

Son of SIMNET born, named BDS-D. Training Electronics & C4I, Vol. 2, No. 4, February 25, 1991.

Pope, A., 1989

The SIMNET Network and Protocols: Version 6. BBN Report 7102, 1989.

Powell, M., Hare, D., Waldo, J., Sventek, J., 1991
The Distributed Object Management Facility: An Architecture for Distributed Objects.

Pullen, J., Entzminger, J., 1991
Applications of Networking Technology in Computer-Assisted Exercises.
In Proceedings of MILCOM 1991.

Soley, Richard M. (Ed)., 1990
Object Management Architecture Guide 1.0. Object Management Group Document 90.9.1, November 1990.

Stroustrup, B., 1988
What is object-oriented programming? IEEE Software, May 1988.

Sun/Hewlett-Packard, 1991.

SunSoft, 1991a
Solaris Open Windows: Introduction to the ToolTalk Service.
Sun Microsystems, 1991.

SunSoft, 1991b
Solaris Open Windows: ToolTalk in Electronic Design Automation.
Sun Microsystems, 1991.

SunSoft, 1991c
Project DOE: Distributed Objects Everywhere.
Sun Microsystems, 1991.

SunSoft, 1991d
Distributed Objects Everywhere: A New Vision for Computing.
Sun Microsystems, 1991.

Weatherley, R., Seidel, D., Weissman, J., 1991

Aggregate Level Simulation Protocol. Summer Computer Simulation Conference, July 1991.